Sleeping on the Job: Energy-Efficient and Robust Broadcast for Radio Networks

Valerie King* Cynthia Phillips† Jared Saia‡ Maxwell Young §

Abstract

We address the problem of minimizing power consumption when broadcasting a message from one node to all the other nodes in a radio network. To enable power savings for such a problem, we introduce a compelling new data streaming problem which we call the *Bad Santa problem*. Our results on this problem apply for any situation where: 1) a node can listen to a set of n nodes, out of which at least half are non-faulty and know the correct message; and 2) each of these n nodes sends according to some predetermined schedule which assigns each of them its own unique time slot. In this situation, we show that in order to receive the correct message with probability 1, it is necessary and sufficient for the listening node to listen to a $\Theta(\sqrt{n})$ expected number of time slots. Moreover, if we allow for repetitions of transmissions so that each sending node sends the message $O(\log^* n)$ times (i.e. in $O(\log^* n)$ rounds each consisting of the n time slots), then listening to $O(\log^* n)$ expected number of time slots suffices. We show that this is near optimal.

We describe an application of our result to the popular grid model for a radio network. Each node in the network is located on a point in a two dimensional grid, and whenever a node sends a message m, all awake nodes within L_∞ distance r receive m. In this model, up to $t < \frac{r}{2}(2r+1)$ nodes within any $2r+1$ by $2r+1$ square in the grid can suffer Byzantine faults. Moreover, we assume that the nodes that suffer Byzantine faults are chosen and controlled by an adversary that knows everything except for the random bits of each non-faulty node. This type of adversary models worst-case behavior due to malicious attacks on the network; mobile nodes moving around in the network; or static nodes losing power or ceasing to function. Let $n = r(2r+1)$. We show how to solve the broadcast problem in this model with each node sending and receiving an expected $O(n \log^2 |m| + \sqrt{n}|m|)$ bits where $|m|$ is the number of bits in m, and, after broadcasting a fingerprint of m, each node is awake only an expected $O(\sqrt{n})$ time slots. Moreover, for $t \leq (1-\epsilon)(r/2)(2r+1)$, for any constant $\epsilon > 0$, we can achieve an even better energy savings. In particular, if we allow each node to send $O(\log^* n)$ times, we achieve reliable broadcast with each node sending $O(n \log^2 |m| + (\log^* n)|m|)$ bits and receiving an expected $O(n \log^2 |m| + (\log^* n)|m|)$ bits and, after broadcasting a fingerprint of m, each node is awake for only an expected $O(\log^* n)$ time slots. Our results compare favorably with previous protocols that required each node to send $\Theta(|m|)$ bits, receive $\Theta(n|m|)$ bits and be awake for $\Theta(n)$ time slots.

*Department of Computer Science, University of Victoria, Victoria, BC, Canada, email: val@cs.uvic.ca

†Sandia National Laboratories, Albuquerque, NM, USA, email: caphill@sandia.gov

‡Department of Computer Science, University of New Mexico, Albuquerque, NM, USA, email: saia@cs.unm.edu

§David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, Canada, email: m22young@cs.uwaterloo.ca. Part of this work was done while a student at the University of New Mexico.

1 Introduction

Power is one of the most critical resources in radio networks. The wireless network cards on radio network devices offer a number of different modes typically with states such as *off, sleeping, idle, receiving* and *sending* [1]. The energy costs across these modes can vary significantly. Remarkably, the cost of the idle, receiving, and sending states are roughly equivalent, and these costs are an order of magnitude larger than the cost of the sleep state.[1] Thus, to a first approximation, the amount of time spent in the sleep state gives an excellent estimate of the energy efficiency of a given algorithm [4]. In this paper, we consider a node to be either asleep or awake (listening and/or sending). Here, our goal is to design an algorithm that allows a single node to broadcast a message so that eventually all non-faulty nodes learn the correct message; this is the problem of reliable broadcast. All previous work on the reliable broadcast problem ignores energy efficiency, assuming the nodes are spending a substantial amount of time listening. In this paper, we directly address the problem of designing energy-efficient algorithms for reliable broadcast; our approach depends upon the analysis of a new data streaming problem that we call the Bad Santa problem.

1.1 The Bad Santa Problem

Consider the following scenario. A child is presented with n boxes, one after another. When given each box, the child must immediately decide whether or not to open it. If the child decides not to open a box, he is never allowed to revisit it. At least half the boxes have presents in them, but the decision as to which boxes have presents is made by an adversarial Santa who wants the child to open as many empty boxes as possible. The child wants to find a present, while opening the smallest expected number of boxes. This is the *Bad Santa problem*.

More formally, an adversary sends a stream of n bits of which at least half are 1. The adversary sets the bits of the stream prior to sending the first bit. The algorithm may query any bit as it passes, but once a bit passes without being queried, it is lost. The algorithm is correct if it always finds a 1. The adversary knows the (randomized) algorithm ahead of time but not its random bits. The cost of an algorithm on an input is the number of expected queries executed until it finds a 1. The goal is to design a correct algorithm with minimum expected cost over the worst case input. At first glance, it may appear that randomly sampling $O(\log n)$ presents trivially solves the single stream Bad Santa problem. However, this strategy has a (small) probability of failure, which is unacceptable.

We are interested in two variants of this problem. First is the single stream case described above. Second is the multi-stream case where there are multiple n-bit streams that the algorithm queries consecutively. Each stream has a constant fraction of 1 bits, but the values (1s and 0s) may be distributed differently in each stream; note, in the multi-stream case, the fraction of 1 bits can be less than $1/2$. A correct algorithm must find one 1 bit in one of the streams. The cost is the expected number of queries over the worst case set of such streams.

1.2 Reliable Broadcast Grid Model

We demonstrate the applicability of the Bad Santa problem on a network model that has been studied extensively in the distributed computing literature, which we will refer to as the *reliable radio broadcast grid model* [5, 6, 7, 8]. In this model, each node is situated on a point in a two-dimensional grid. Whenever a node sends a message, all awake nodes within L_∞ distance r receive the message.[2] Communication is synchronous. If two nodes broadcast simultaneously, the messages interfere, so nodes in the intersection of the neighborhoods of both senders receive no message. We make some additional remarks regarding the flexibility of the grid model later on in Section 5.5.

[1] The difference in energy consumption between the idle/send/receive states and the sleep state differs depending on the type of card and the communication standard being employed. For example, using the IEEE 802.11 standard with a 11 Mbps card, the ratios between power consumption of the idle/send/receive states and the sleep state are all more than 12 [2]. In [3], with a different setup employing TinyOS and a TR1000 transceiver, the measured ratios are over 1000.

[2] The distance between two points (x_1, y_1) and (x_2, y_2) in the L_∞ metric is $\max\{|x_1 - x_2|, |y_1 - y_2|\}$. We can use other metrics; the choice determines the fraction of faulty nodes we can tolerate in any $(2r + 1) \times (2r + 1)$ square of the grid.

1.2.1 Faults

Every node in the grid may suffer faults, but as in [5, 6, 7, 8] we assume that no more than t nodes in any $2r+1$ by $2r+1$ square are faulty and that no node can spoof another node's identity. We consider the cases where these faults are either all *fail-stop*: the t nodes are all deleted from the network; or *Byzantine*: the t nodes are taken over by an adversary and deviate from our protocol arbitrarily.[3] We assume that all of the nodes that suffer faults are chosen by a single adversary who controls these nodes to coordinate attacks on the network. This adversary knows everything except for the random bits of the non-faulty nodes.

1.2.2 Schedule of Transmission

We assume there is a distinguished node s known as the *source* that holds an initial message m. We assume without loss of generality that the source node has coordinates $(0,0)$ on the grid, i.e. *all nodes know the source*. We discuss relaxing this assumption in Section 2.2. All known protocols designed for the reliable broadcast grid model proceed in steps where the source of the message sends to its neighbors, which in turn send to their neighbors, until all nodes receive the message. The predecessor set G_p of a correct node p is a particular set of nodes such that if p listens to all nodes in G_p and majority filters on the received messages, p will obtain the correct message; we give a precise definition of G_p in Section 5. Following the literature, we assume that each node has a predecessor set of $n = r(2r+1)$ nodes assigned to distinct time slots and that the entire schedule repeats every $(2r+1)^2$ time slots. We call each schedule repetition a *round*. An example of a broadcast schedule is given in [8]: In each round, each node in position (x, y) broadcasts in time slot $((x \mod (2r+1)) \times (2r+1) + (y \mod (2r+1))) \mod (2r+1)^2$. For the remainder of the paper, it suffices to assume each round has $O(n)$ time slots and each node within L_∞ distance r of some node p is assigned to a distinct time slot. If $t < n/2$ then each node has a predecessor set of which strictly less than half of the nodes are faulty, or it can listen directly to the source which we assume is correct [5, 6]. For simplicity, we assume the source initially broadcasts the message size and, thereafter, time slots are long enough to send the entire message.[4] The cost of listening to a message is proportional to the message length.

2 The Bad Santa Problem & The Reliable Broadcast Problem

In this section, we sketch the methods for applying the solutions of the Bad Santa problem to the problem of reliable broadcast. We will reduce the expected listening time (i.e. number of time slots in an awake state) and the expected bit complexity required for a node to learn the message from its n predecessors. We can use the algorithm for the single stream Bad Santa problem to do so *provided* that: (1) at least half of the predecessors have the correct message and, in the case of Byzantine faults, the listening node can determine if a message is correct; (2) the listening node knows the location of the source node and time of broadcast (to determine when to start the Bad Santa protocol and to which set of n nodes to possibly listen). If $t < n/2$; faults are fail-stop; and the time of broadcast and length of the message is known in advance; conditions (1) and (2) are clearly satisfied. Thus, the message can be transmitted safely from one set of predecessors to another, with each node using the Bad Santa protocol to decide which of its predecessors to listen to and thereby learn the message. In this case, there is no change to the latency of the broadcast; each node sends once.

We can reduce listening time further by using the multi-stream Bad Santa protocol. Here the fraction of faulty predecessors can exceed $1/2$ and we show that multiple streams are required if we wish to obtain savings. If we use $k+1$ streams, then there are $k+1$ rounds of sending before the message is passed from one set of nodes to another; each node sends $k+1$ times and the latency increases by a factor of $k+1$ over the single round case.

[3] Although, Byzantine nodes must also abide by the schedule as in [5, 6, 8]

[4] An alternative is that the source node preprocesses the message by dividing it into pieces that each fit into a time slot. However, both the broadcasting of the message size and the details of how the message might be formatted for sending are outside the scope of this paper.

Notation	Definition
r	Radius of broadcast for all nodes.
t	Number of Byzantine peers in a $(2r+1) \times (2r+1)$ square of the radio network.
$p(x, y)$	A node p located at coordinate (x, y) in the grid network model.
$N(p)$ or $N(x, y)$	Set of nodes within the broadcast radius of node $p(x, y)$.
n	In the context of the Bad Santa Problem, n is the number of boxes in a stream. In the context of a radio network, n is the size of predecessor set where $n = r(2r+1)$.
k	Number of streams used in the problem definition of the Bad Santa Problem.
s	Source node (or dealer) in the problem of reliable broadcast.
m	Message sent by the source node in the problem of reliable broadcast.
$\|m\|$	Number of bits in the message m.
f	A secure hash function.
$f(m)$	Fingerprint resulting from applying the hash function f to m.

Table 1: Summary of frequently used notation.

Failure is Not An Option: Why do we insist on allowing no error in the Bad Santa problem? Why not just use random sampling? Random sampling has a probability of error that depends on n, which is on the order of the number of nodes in the transmission radius; we stress that n depends on r and is *not* the total number of nodes in the network. If the network's total size is much larger than n, then even if the failure probability for a single listener is exponentially small in n, the probability that some node in the network fails to learn the message will still be quite large. For example, if the total network size is exponential in n and the probability of failure for a single listener is $O(2^{-n})$, then with constant probability, reliable broadcast will fail.

2.1 Byzantine Fault Model: Known Start Time and Source

To satisfy condition (1) when the faults are Byzantine, our protocol has two stages. In the first stage, the source uses a secure (cryptographic) hash function (for more on such hash functions see [9], Chapter 4) to generate a fingerprint of size $(\log_2 |m|)^2$ where $|m|$ is the message length[5], and broadcasts this fingerprint to all the other nodes in the network using a previously known energy-inefficient method in [6]. In the second stage, the source broadcasts the full message with each node using a Bad Santa protocol. Each node compares the hash value of each full message received against the true fingerprint to determine if it agrees and is thus presumably correct. If the adversary is unable to discover a false message whose hash matches the fingerprint, then the only message which matches the fingerprint is the correct message. Each node can determine if the message it receives is correct. Thus, at each stage, all non-faulty nodes transmit the correct message and condition (1) is satisfied. This introduces a possibility of error into the transmission which depends on the relative size of the fingerprint to the message and the resources of the adversary. In this model, the set of faulty nodes can differ from one stream to the next as chosen by the adversary; however, for a given stream, the adversary must decide whether a node is corrupt prior to its selection or non-selection by a protocol.

2.2 Byzantine Fault Model: Unknown Start Time and Source(s)

We also deal with the case where the start time of the message is not known in advance, or the location of the source is not known. Moreover, our protocol allows any node to send a message i.e. become a source node. We note that this is also possible under the original protocols of [5, 6, 8]; however, we explicitly deal with this case and show how to accomplish an energy savings if $t < \frac{n}{16+\epsilon}$ for any constant $\epsilon > 0$. More specifically, we require that no more than a $1/2 - \epsilon$ fraction of the nodes are faulty in any $r/2$ by $r/2$ square. In this model, the adversary is adaptive in the sense that it can decide which nodes to take over based on which nodes have previously committed to the correct message.

[5] We make the random oracle assumption about the hash function used to generate the fingerprint of m.

3 Our Results

Our five main results are summarized in the theorems below. Theorem 1 is given in Section 4; Theorem 2 in Section 4.1; Theorem 3 and Theorem 4 are addressed in Sections 5.1& 5.2; Theorem 5 and Theorem 6 in Section 5.3; Theorem 7 in Section 5.4. For ease of exposition, we have aggregated the notation we most commonly use throughout the paper in Table 1. Finally, throughout, let $\lg n$ denote the logarithm base 2 and let $\log^{(k)} n$ denote $\underbrace{\log \cdots \log}_{k} n$.

Theorem 1. *For the single stream Bad Santa problem, the optimal expected number of queries is* $\Theta(\sqrt{n})$.

Theorem 2. *For the k stream Bad Santa problem, the optimal expected number of queries is* $O(\log^{(k)}(n) + k)$ *and* $\Omega(\log^{(2k)} n)$. *In particular, for* $k = \Theta(\log^* n)$, *we can ensure the expected number of queries is* $O(\log^* n)$.

The next two theorems about energy-efficient broadcast are established by algorithms based on solutions to the Bad Santa problem. We again repeat that $n = r(2r+1)$ and so n depends on the broadcast radius; it is not the total number of nodes in the network. The algorithms apply to a grid of finite or infinite size. In the former case, we achieve the standard result that all nodes, except those on the boundary of width r, commit to the correct message. In the latter case, for Byzantine faults, our result translates into a finite portion of the grid obtaining the correct message and this is dependent on the computational power of the adversary. Theorem 3 essentially follows directly from Theorems 1 and 2. Theorem 4 requires a fingerprint of the message to first be broadcast through the network.

Theorem 3. *Assume we have a network where at most* $t < \frac{r}{2}(2r+1)$ *nodes suffer fail-stop faults in any square of size* $2r+1$ *by* $2r+1$ *and that the start time and source of a message are known. Then there exists an algorithm for reliable broadcast which has the following properties:*

- *Each node is awake for* $O(\sqrt{n})$ *time slots in expectation.*
- *Each node broadcasts* $O(\sqrt{n}|m|)$ *bits and receives* $|m|$ *bits.*

In the next theorem, we use the notion of *computational steps* in the context of the adversary. By this, we mean the number of times the adversary can create an input x', apply a secure hash function f to x' and check for a match between the output fingerprint $f(x')$ and some other fingerprint for which the adversary is attempting generate a collision.

Theorem 4. *Assume we have a network where at most* $t < \frac{r}{2}(2r+1)$ *of the nodes suffer Byzantine faults in any square of size* $2r+1$ *by* $2r+1$ *and that the start time and source of a message are known. Further assume that the number of computational steps available to the adversary is bounded by s. Then there exists an algorithm for guaranteeing reliable broadcast with a probability of failure* $O(s/|m|^{\lg|m|})$. *In an initial stage, the algorithm requires a fingerprint of size* $\lg^2|m|$ *to be initially broadcast to the network. However, in the second stage, when the message m itself is broadcast, the algorithm has the following properties:*

- *Each node is awake for* $O(\sqrt{n})$ *time slots in expectation,*

Over both stages, the algorithm has the following costs:

- *Each node broadcasts* $O(n\log^2|m| + \sqrt{n}|m|)$ *bits and receives an expected* $O(n\log^2|m| + \sqrt{n}|m|)$ *bits.*

We also present results on increased energy-savings for values of t within an arbitrary constant factor of optimal. In particular, we consider the case where $t \leq (1-\epsilon)\frac{r}{2}(2r+1)$ for any constant $\epsilon > 0$ where we have the following results:

Theorem 5. *Assume we have a network where, for any constant* $\epsilon > 0$, *at most* $t \leq (1-\epsilon)\frac{r}{2}(2r+1)$ *nodes suffer fail-stop faults in any square of size* $2r+1$ *by* $2r+1$ *and that the start time and source of a message are known. Then there exists an algorithm which guarantees reliable broadcast and which has the following properties:*

- *For any k between 1 and $\ln^* n$, the algorithm requires each node to be awake for an expected $O(\log^{(k)} n)$ time slots.*
- *Each node broadcasts $O(k|m|)$ bits and receives $|m|$ bits.*

Therefore, the above algorithm requires each node to broadcast $O(k)$ times which translates into a higher latency given that nodes must adhere to a broadcast schedule; however, nodes expend far more energy in expectation.

Theorem 6. *Assume we have a network where, for any constant $\epsilon > 0$, at most $t \leq (1-\epsilon)\frac{r}{2}(2r+1)$ of the nodes suffer Byzantine faults in any square of size $2r+1$ by $2r+1$ and that the start time and source of a message are known. Further assume that the number of computational steps available to the adversary is bounded by s. Then there exists an algorithm which guarantees reliable broadcast with a probability of failure $O(s/|m|^{\lg|m|})$. In an initial stage, the algorithm requires all nodes to be awake for every slot during which a fingerprint of size $\lg^2|m|$ is initially broadcast to the network. However, in the second stage, when the message m itself is broadcast, the algorithm has the following properties:*

- *For any k between 1 and $\ln^* n$, requires all nodes to be awake an expected $O(\log^{(k)} n)$ time slots.*

Over both stages, the algorithm has the following costs:

- *For any k between 1 and $\ln^* n$, each node broadcasts $O(n\log^2|m| + k|m|)$ bits and receives an expected $O(n\log^2|m| + (\log^{(k)} n)|m|)$ bits.*

Finally, we deal with the case where the start time and the source of the message is unknown. In this situation, if $t < \frac{n}{16+\epsilon}$, we have the following result:

Theorem 7. *If the start time and source of a message are unknown, there is a protocol for reliable broadcast in which each node (1) sends $O(|m|)$ bits per round, (2) is awake an amortized constant number of time slots per round and (3) receives an amortized $O(|m|)$ bits per round.*

For this last result given in Theorem 7, all nodes may receive the message; that is, those nodes on the boundary are not excluded as with our previous results.

To contrast our results with previous work, we note that under the previous algorithms for reliable broadcast [5, 6], each node 1) is awake for $(2t+1) = \Theta(n)$ time slots, 2) broadcasts $\Theta(|m|)$ bits; 3) receives $\Theta(|m|)$ bits in the fail-stop model; and 4) can be forced by the adversary to receive $\Theta(n|m|)$ bits in the Byzantine fault model. Therefore, in both fault models, our algorithms are saving substantially on the amount of time a node must be awake for listening to the full message. For the fail-stop case, for $k \geq 1$, we are trading a small factor increase in traffic for these savings . Moreover, in the Byzantine case, we greatly reduce the total bit complexity. Finally, note that $|m|$ need not be large to make the probability of finding a message with the same fingerprint very small. For example, if $|m| = 1$ kB, the probability of a collision is already less than 10^{-30}.

3.1 Related Work

The reliable broadcast problem over the radio network model described above has been extensively studied in [5, 6, 7, 8, 10]. In [8], Koo showed that reliable broadcast with Byzantine faults is impossible if $t \geq \frac{r}{2}(2r+1)$ in the L_∞ norm. In [5, 6], Bhandari and Vaidya presented a clever algorithm that achieved reliable broadcast tolerating Byzantine faults for any $t < \frac{r}{2}(2r+1)$; our Theorem 4 appies to this scenario. There the authors also achieve $t < r(2r+1)$ for the fail-stop fault model whereas our result applies only when $t < \frac{r}{2}(2r+1)$ or when $t \leq (1-\epsilon)(r/2)(2r+1)$ for any constant $\epsilon > 0$. Therefore, we are a constant factor from the optimal tolerance in the fail-stop model. Koo et al., in [7], described an algorithm that achieves reliable broadcast even when the faulty nodes can spoof addresses of honest nodes or cause collisions; this is a more challenging fault model than is addressed in our work or in any other previous work. All prior algorithms proposed for the reliable broadcast problem require each node in the network to be awake for a constant fraction of the time slots and thus are not energy-efficient. Our algorithm from Theorem 4 makes

use of the algorithm from [6] to broadcast a fingerprint of the message. Finally, under different models of a radio network, the problems of consensus [11, 12], reliable broadcast under the fail-stop fault model [13] and reliable broadcast under adversarial faults [14] have been studied. Work in [15] deals with broadcast protocols in a time-slotted network where the number of times a node can transmit is constrained; this is called "k-shot broadcasting". The authors focus on establishing bounds on the number of rounds each node must transmit in order to achieve broadcast; hence, there is a focus on the tradeoff between energy (i.e. the number of shots needed) and latency of the broadcast. However, despite this similarity, the network model used in [15] does not incorporate adversarial behaviour and captures more general topologies; consequently, the techniques and results differ significantly from our work.

Data streaming problems have been popular in the last several years [16, 17]. Generally, past work in this area focuses on computing statistics on the data using a small number of passes over the data stream. In [16], the authors treat their data stream as a directed multi-graph and examine the space requirements of computing certain properties regarding node degree and connectedness. Munro and Paterson [18] consider the problem of selection and sorting with a limited number of passes over one-way read-only memory. Guha and McGregor [19, 20] examine the problem of computing statistics over data streams where the data objects are ordered either randomly or arbitrarily. Alon, Matias and Szegedy [21] examine the space complexity of approximating the frequency of moments with a single pass over a data stream. In all of these cases, and others [22, 23], the models differ substantially from our proposed data streaming problem. Rather than computing statistics or selection problems, we are concerned with the guaranteed discovery of a particular value, and under our model, expected query complexity takes priority over space complexity.

A preliminary version of the results in this paper appeared in [24]. This current version contains a complete description of our protocols along with the full proofs of our results. We also correct an error regarding the energy-savings achieved in [24]. That is, in the case where $t < (r/2)(2r+1)$, we achieve what is essentially a quadratic reduction in resource costs. In the process of amending our results, we treat the case for $t \leq (1-\epsilon)(r/2)(2r+1)$ for any constant $\epsilon > 0$, which yields the further energy savings identical to those previously claimed; this also improves on our previous result for the fail-stop model. An extended discussion of previous results is also provided along with a examination of certain practical considerations regarding the utility of the algorithms we are proposing.

4 Single Stream Bad Santa

We now consider the single stream Bad Santa problem. A naive algorithm is to query $n/2+1$ bits uniformly at random. The expected cost for this algorithm is $\Theta(n)$ since the adversary will place the 1's at the end of the stream. The following is an improved algorithm.

Single Stream Strategy

1. Perform $\sqrt{n}$ queries uniformly at random from the first half of the stream. Stop immediately upon finding a 1.
2. If no 1 has been found, starting with the first bit in the second half of the stream, query each consecutive bit until a 1 is obtained.

Lemma 1. *The expected cost of the above strategy is $O(\sqrt{n})$.*

Proof. Assume that there are $i\sqrt{n}$ 1s in the first half of the stream where $i \in [0, \frac{\sqrt{n}}{2}]$. This implies that there are then $(n/2) - i\sqrt{n}$ 1s in the second half of the stream. By querying $\sqrt{n}$ slots uniformly at random in the first half of the stream, the probability that the algorithm fails to obtain a 1 in the first half is no more than:

$$\left(1 - \frac{i\sqrt{n}}{(n/2)}\right)^{\sqrt{n}} = \left(1 - \frac{2i}{\sqrt{n}}\right)^{\sqrt{n}}$$

for an expected overall cost not exceeding:

$$\sqrt{n} + \left(1 - \frac{2i}{\sqrt{n}}\right)^{\sqrt{n}} \cdot i\sqrt{n}.$$

We find the maximum by taking the derivative:

$$\frac{d}{di}\left(1 - \frac{2i}{\sqrt{n}}\right)^{\sqrt{n}} \cdot i\sqrt{n} = \sqrt{n}\left(1 - \frac{2i}{\sqrt{n}}\right)^{\sqrt{n}} - 2i\sqrt{n}\left(1 - \frac{2i}{\sqrt{n}}\right)^{\sqrt{n}-1}$$

and setting it to zero while solving for i gives $i = \frac{\sqrt{n}}{2(\sqrt{n}+1)}$. Plugging this into the expected cost function gives an expected cost of $O(\sqrt{n})$. □

We now show that this bound is optimal to within a constant factor. In the proof of the following, let $\tilde{O}$ denote that logarithmic factors are ignored.

Lemma 2. *$\Omega(\sqrt{n})$ expected queries are necessary in the single stream case.*

Proof. We follow Yao's min-max method [25] to prove lower bounds on any randomized algorithm that errs with probability no greater than $\lambda = 1/2^{\tilde{O}(\sqrt{n})}$: We describe an input distribution and show that any deterministic algorithm that errs with tolerance (average error) less than $2\lambda = 1/2^{\tilde{O}(\sqrt{n})}$ on this input distribution requires $\Omega(\sqrt{n})$ queries on average for this distribution. By [25], this implies that the complexity of any randomized algorithm with error λ has cost $(1/2)\Omega(\sqrt{n}) = \Omega(\sqrt{n})$. Let $[a, b]$ denote the bits in position $a, a+1, ..., b-1, b$ of the stream. The distribution is as follows:

CASE 1. With probability $1/2$, $\sqrt{n}$ uniformly distributed random bits in $[1, n/2]$ are set to 1 and the remaining bits in that interval are 0, $[n/2+1, n/2+\sqrt{n}]$ are all set to 0, and the remaining bits are 1.

CASE 2.x: For x= $0, ..., \sqrt{n}-1$, with probability $1/(2\sqrt{n})$, $[1, ..., n/2]$ contains a uniformly distributed random set of x 0's and the rest are 1's; $[n/2+1, n/2+\sqrt{n}]$ contains a uniformly distributed random set of x 1's and the rest are 0's; and the remaining bits in the stream are 0.

Analysis: Let A be a deterministic algorithm which errs with average probability less than 2λ. Note that A is completely specified by a list L of indices of bits to query while it has not yet discovered a 1, since it stops as soon as it sees a 1. Let x be the number of queries in the list that lie in $[1, n/2]$. For a constant fraction of inputs in CASE 1, A will not find a 1 in $[1, n/2]$ within $\sqrt{n}$ queries. Hence either $x \geq \sqrt{n}$ or A must find a 1 with high probability in $[n/2+1, n]$. Now suppose $x < \sqrt{n}$. We show that A's list L must contain greater than $\sqrt{n} - x$ bit positions in $[n/2+1, n/2+\sqrt{n}]$. To show this, assume this is untrue. Then A will err on the input in CASE 2.x in which all the x positions queried in $[1, n/2]$ and the $\sqrt{n} - x$ positions queried in $[n/2+1, n/2+\sqrt{n}]$ are 0. Note that this input occurs with probability $(2\sqrt{n})^{-1}\binom{n/2}{x}^{-1}\binom{\sqrt{n}}{x}^{-1} \geq 2\lambda$ in the distribution. Therefore, the algorithm errs with probability at least 2λ; this is a contradiction. We conclude that any algorithm erring with probability less than 2λ must either have $x \geq \sqrt{n}$ or queries greater than $\sqrt{n} - x$ bits of $[n/2+1, n/2+\sqrt{n}]$.

Now we show that any such deterministic algorithm incurs an average cost of $\Omega(\sqrt{n})$ on the CASE 1 strings in this distribution. If $x \geq \sqrt{n}$ then for a constant fraction of strings in CASE 1, the algorithm will ask at least $\sqrt{n}$ queries in $[1, n/2]$ without finding a 1. If $x < \sqrt{n}$, then with constant probability the algorithm will incur a cost of x in $[1, ..n/2]$ and go on to incur a cost of $\sqrt{n} - x$ in $[n/2+1, n/2+\sqrt{n}]$ since all the values there are 0. Therefore, we have shown that the distributional complexity with error 2λ is $\Omega(\sqrt{n})$. It follows from [25] that the randomized complexity with error λ is $\Omega(\sqrt{n})$. □

Theorem 1 follows immediately from Lemma 1 and Lemma 2. We finish this section by showing that if the fraction of 1s in the single stream case, δ, is less than $1/2$, then the number of expected queries is $\Omega(n)$. The proof is very similar to that of Theorem 2.

Theorem 8. *For the single round dynamic Bad Santa problem with $\delta = \frac{1}{2} - \epsilon$, the number of queries in expectation is $\Omega(n)$ for an arbitrarily small constant $\epsilon > 0$.*

Proof. We apply the min-max method of [25] to prove lower bounds on any randomized algorithm that fails with probability no greater than $\lambda = 2^{-\Theta(n \log n)}$. An input distribution is used to show that any deterministic algorithm failing with probability less than $2\lambda = 2^{-\Theta(n \log n)}$ on this input distribution requires $\Omega(n)$ queries. By [25], this implies that the complexity of any randomized algorithm with error λ has cost $\Omega(n)$. Let $[a, b]$ denote the bits in position $a, a+1, ..., b-1, b$ of the stream. Our input distribution is as follows:

- *Case* 1*:* With probability 1/2, a constant number of bits, c, chosen uniformly at random without replacement in $[1, \delta n]$ are set to 1 and the remaining bits in that interval are 0. In $[\delta n+1, (1-\delta)n+c]$ all bits are set to 0, and the remaining bits are set to 1.

- *Case* 2.x*:* For x= $0, ..., \delta n - 1$, with probability $1/(2\delta n)$, $[1, \delta n]$ contains a uniformly distributed random set (without replacement) of x 0s and the rest are 1s; $[\delta n + 1, (1 - \delta)n + c]$ contains a uniformly distributed random set (without replacement) of x 1s and the rest are 0s. The remaining bits in the stream are 0.

Cost Analysis: Let $\mathcal{A}$ be a deterministic algorithm which fails with probability less than 2λ. Note that $\mathcal{A}$ is completely specified by a list L of indices of bits to query while it has not yet discovered a 1. Let y be the number of queries in the list that lie in $[1, \delta n]$. With constant probability $\mathcal{A}$ will fail to find a 1 in $[1, \delta n]$ within $(\delta - \epsilon')n$ queries where $\epsilon' > 0$ is an arbitrarily small constant. This is because sampling without replacement, the probability that $\mathcal{A}$ fails is $\prod_{i=1}^{(\delta-\epsilon')n} \left(1 - \frac{c}{(\delta n)-i}\right) \geq \left(1 - \frac{c}{\epsilon' n}\right)^{(\delta-\epsilon')n} = \Theta(1)$. Therefore, either $y \geq (\delta - \epsilon')n$ or A must find a 1 with high probability in $[\delta n + 1, n]$. Now suppose $y < (\delta - \epsilon')n$. We show that L must contain greater than $(1-2\delta)n + c - y$ bit positions in $[\delta n + 1, (1-\delta)n + c]$. To show this, assume this is untrue. Then $\mathcal{A}$ will fail on the input in Case 2x in which all the y positions queried in $[1, \delta n]$ and the $(1-2\delta)n + c - y$ positions queried in $[\delta n + 1, (1-\delta)n + c]$ are 0. Note that this input occurs with probability $\binom{1}{2\delta n}\binom{\delta n}{y}^{-1}\binom{(1-2\delta)n+c}{(1-2\delta)n+c-y}^{-1} = \binom{1}{2\delta n}\binom{\delta n}{y}^{-1}\binom{(1-2\delta)n+c}{y}^{-1} \geq \binom{1}{2\delta n}\left(\frac{y^2}{\delta n e^2((1-2\delta)+c)}\right)^y \geq 2\lambda$ for $y = 0, ..., (\delta - \epsilon')n - 1$ and for sufficiently large n. Therefore, the algorithm fails with probability at least 2λ which is a contradiction. We conclude that any algorithm failing with probability less than 2λ must either have $y \geq (\delta - \epsilon')n$ or queries greater than $(1-2\delta)n + c - y$ bits in $[\delta n + 1, (1-\delta)n + c]$.

Finally, we can prove the average cost that any such deterministic algorithm incurs on the Case 1 strings in our distribution. As we saw above, if $y \geq (\delta - \epsilon')n$ then for a constant fraction of strings in Case 1, the algorithm will ask at least $(\delta - \epsilon')n$ queries in $[1, \delta n]$ without finding a 1. Else, if $y < (\delta - \epsilon')n$, then with constant probability the algorithm will incur a cost of y in $[1, \delta n]$ and go on to incur a cost of at least $(1-2\delta)n + c - y$ in $[\delta n + 1, (1-\delta)n + c]$ since all the values there are 0; regardless, the cost is $\Omega(n)$ (note that this is not the case if $\delta \geq 1/2$). Therefore, the distributional complexity with error 2λ is $\Omega(n)$. It follows directly from [25] that the randomized complexity is $\Omega(n)$. □

While the optimal expected cost for the single stream is $\Theta(n)$, it is still possible to obtain asymptotic savings over multiple streams when $\delta < 1/2$ and we address this in the next section.

4.1 The Multiple Streams Bad Santa Problem

We define a (α, β)-strategy to be an algorithm which occurs over no more than α streams, each with at least a (possibly different) set of at least $\Theta(n)$ values of 1, and which incurs expected cost (number of queries) at most β. To be explicit, for multiple streams, we can handle the case where the fraction of boxes that contain a 1, denoted by δ, can be less than 1/2. The previous section demonstrated a $(1, O(\sqrt{n}))$-strategy. We now consider the following protocol over $(k+1)$ streams.

Multi-Stream Selection Strategy

For $i = k$ to 1

- Perform $\frac{1}{\delta}\ln^{(i)}(n)$ queries uniformly at random over the entire stream. Stop if a 1 is obtained.

If no value of 1 has been found, then if $\delta \geq 1/2$, use the single stream strategy on the final stream. Otherwise, for $\delta < 1/2$, open each of the n boxes in order in the final stream until a 1 is located.

Lemma 3. *For a constant δ, the above protocol is a $(k+1, O(\log^{(k)}(n)+k))$-strategy.*

Proof. Correctness is clear because in the worst case, we use the correct the single stream strategy, or open all boxes, in the final stream. The expected cost is:

$$\begin{aligned}
&\leq \quad \delta^{-1}\ln^{(k)} n + \left[\sum_{i=k-1}^{1} (1-\delta)^{\delta^{-1}\ln^{(i+1)} n} \cdot O\left(\delta^{-1}\ln^{(i)} n\right)\right] + (1-\delta)^{\delta^{-1}\ln n}\cdot O(n)\\
&\leq \quad \delta^{-1}\ln^{(k)} n + \left[\sum_{i=k-1}^{1} e^{-\ln^{(k)} n} \cdot O\left(\delta^{-1}\ln^{(k-1)} n\right)\right] + e^{-\ln n}\cdot O(n)\\
&= \quad O(\log^{(k)}(n)+k)
\end{aligned}$$

□

Lemma 4. *If there are $\ln^*(n)+1$ streams and δ is a constant, then the multi-stream algorithm provides a $(O(\log^* n), O(\log^* n))$-strategy.*

Proof. By the definition of the iterated logarithm:

$$\ln^* n = \begin{cases} 0 & \text{for } n \leq 1 \\ 1+\ln^*(\ln n) & \text{for } n > 1 \end{cases}$$

if $k = \ln^* n$, we can plug this value into the last line of the proof of Lemma 3 which contains two terms inside the big-O notation. The first term is $1/\delta$, by definition of $\ln^* n$, and the second is $O(\ln^* n)$, for a total expected cost of $O(\ln^* n)$. □

4.2 Lower bound for multiple streams

First, we show the following lemma. For ease of exposition, we assume $\delta = 1/2$; however, any constant δ will suffice with little modification to the proof:

Lemma 5. *$\Omega(\log^{(i+2)} n)$ expected queries are required for a randomized algorithm that errs with probability less than $\lambda = (\ln^{(i)} n)^{-\epsilon}$ on one stream of length n. In particular, when $i = 0$, $\Omega(\log\log n)$ expected queries are required for a randomized algorithm with error less than $1/n^{\epsilon}$, for any constant $\epsilon > 0$.*

Proof. We apply Yao's min-max method [25] and consider the distribution in which with probability 1/3, one of the $I_1 = [1, n/3]$, $I_2 = [n/3+1, 2n/3]$, and $I_3 = [2n/3+1, n]$ intervals is all 0's, and the other two each contain exactly $n/4$ 1's with the 1's distributed uniformly at random. Let L denote the list of queries of a deterministic algorithm, and let x_i be the number of queries in $L \cap I_i$. The probability that the algorithm fails to find a 1 in any interval I_i is $\binom{n/3-x_i}{n/4}/\binom{n/3}{n/4} = \frac{n/12}{n/3}\frac{n/12-1}{n/3-1}\cdots\frac{n/12-x_i+1}{n/3-x_i+1} > \left(\frac{n/12-x_i+1}{n/3-x_i+1}\right)^{x_i} > (\frac{1}{4}-\frac{3x_i}{n})^{x_i} > (\frac{1}{4}-\epsilon)^{x_i} > (\frac{1}{e^{7/4}})^{x_i} = e^{-7x_i/4}$ when $x_i = o(n)$ for sufficiently large n. Let I_i and I_j be the intervals that are not all 0's. Then the probability of failing to find a 1 in either I_i and I_j is $> e^{-7(x_i+x_j)/4}$ for sufficiently large n when $x_i + x_j = o(n)$. Hence the probability of not finding a 1 over all intervals is $> (1/3)e^{-7(x_i+x_j)/4} > 2\lambda$ if $x_i + x_j < (3/7)\epsilon \lg^{(i+1)} n$. We conclude that a deterministic algorithm with average error less than 2λ can have at most one $x_i, i = 1,2,3$ such that $x_i < (3/14)\epsilon \lg^{(i+1)} n$.

Now we examine the cost of such an algorithm. Suppose $x_1 \geq (3\epsilon/14)(\ln^{(i+2)} n)$ then with probability 1/3 I_1 is all 0's and the cost incurred is x_1, for an average cost of $(\epsilon/14)(\ln^{(i+2)} n)$. Now suppose $x_1 < (3\epsilon/14)\ln^{(i+2)} n$. From above, we know $x_2 > (3\epsilon/14)\ln^{(i+1)} n)$. Then with probability 1/3, I_2 is all 0's and with probability $> e^{-7x_1/4} > (\ln^{(i+1)} n)^{-3\epsilon/8}$, the algorithm does not find a 1 in I_1 and incurs a cost of

$(3\epsilon/14)\lg^{(i+1)} n$ in I_2 for an average cost of at least $(\epsilon/14)(\ln^{(i+1)} n)^{1-3\epsilon/8}$. Hence the average cost of any such deterministic algorithm is at least $\min\{(\epsilon/14)(\ln^{(i+2)} n),\ (\epsilon/14)(\lg^{(i+1)} n)^{1-3\epsilon/8}\} = \Omega(\ln^{(i+2)} n)$. By Yao's min-max method [25], any randomized algorithm with error λ is bounded below by $1/2$ the average cost of a deterministic algorithm with average error 2λ on any distribution. The lemma now follows. □

Lemma 6. *For $k > 0$, $\Omega(\ln^{(2k)} n)$ expected queries are necessary to find a 1 from $k+1$ streams with probability 1.*

Proof. We use induction on the number of streams:

Base Case: Let $k = 1$. Either the algorithm finds a 1 in the first pass or the second pass. From Lemma , for any constant ϵ any algorithm that fails to find a 1 in the first pass with probability $\leq n^{-\epsilon}$ has expected cost $\Omega(\log\log n)$. If the algorithm fails to find a 1 in the first pass with probability at least $n^{-\epsilon}$ then the expected cost to the algorithm is at least the probability it fails in the first pass times the expected cost of always finding a 1 in the second and final pass, which is $n^{-\epsilon} \cdot \Omega(\sqrt{n})$. (The second factor is from Lemma 2. Choosing $\epsilon < 1/2$, the expected cost is $\Omega(\log\log n)$.

Inductive Hypothesis: For $k > 1$, $\Omega(\ln^{(2k)} n)$ expected queries are necessary to find a 1 from $k+1$ streams with probability 1.

Inductive Step: Now assume the hypothesis is true for up to $k > 1$ streams. Assume we have $k+1$ streams. Any randomized algorithm either fails to find a 1 in the first stream with probability less than $(1/\ln^{(2k-2)} n)^{\epsilon}$, in which case by Lemma 5, the expected cost of the algorithm when it processes the first stream is $\Omega(\ln^{(2k)} n)$ or the probability that it fails in the first pass is at least $(1/\ln^{(2k-2)} n)^{\epsilon}$. In that case, the expected cost deriving from queries of the second stream is at least $(1/\ln^{(2k-2)} n)^{\epsilon} \cdot \Omega(\ln^{(2k-2)} n)$ where the second factor of this expression is the expected number of queries needed to find a 1 in k streams, as given by the induction hypothesis. The minimum expected cost of any randomized algorithm is the minimum of these two possibilities, which is $\Omega(\ln^{(2k)} n)$. □

Theorem 2 then follows immediately from Lemmas 3, 4, 5 and 6.

5 Reliable Broadcast Protocols

We begin by recalling some notation from Table 1 and briefly reviewing the protocol of [6]. Let $p(x, y)$ denote the node p at location (x, y) in the grid. We define a corridor of width $2r + 1$ starting at the source located at point $(0, 0)$ and ending at node $p = (x, y)$. We will use $N(p)$ or $N(x, y)$ to denote the set of nodes within radius r of $q(x, y)$; this is the neighborhood of q. Additionally, we define the perturbed neighborhood $PN(p)$ of $p(a, b)$ as $PN(p) = N(a + 1, b) \cup N(a - 1, b) \cup N(a, b + 1) \cup N(a, b - 1)$. The following protocol for reliable broadcast in the presence of Byzantine faults is by Bhandari and Vaidya [6]. In this protocol, the message COMMIT(i, v) signifies that node i has committed to value v, and the message HEARD(j, i, v) signifies that node j has heard a message COMMIT(i, v).

Reliable Broadcast Protocol (Bhandari and Vaidya, 2005)

- Initially, the source s does a local broadcast of v.
- Each node $i \in N(s)$ commits to the first value it receives from s and does a one-time broadcast of COMMIT(i, v).
- The following protocol is executed by each node j (including those nodes in the previous two steps):
  - On receipt of a COMMIT(i, v) message from a neighbor i, j records the message and broadcasts HEARD(j, i, v).
  - On receipt of a HEARD(j', i, v), j records this message.

– Upon receiving COMMIT or HEARD messages that 1) claim v as the correct value and 2) are received along at least $t+1$ node disjoint paths that all lie within a single neighborhood, then node j commits to v and does a one time broadcast of COMMIT(j, v).

Proving that this protocol is correct is non-trivial and we refer the reader to [6] for details. To briefly summarize, the proof in [6] works by showing that for each node p in $PN(a,b) - N(a,b)$, there exist $2t+1$ paths $P_1, ..., P_{2t+1}$ belonging to a single neighborhood $N(a, b+r+1)$, each having one of the forms listed below:

- $P_i = (q, p)$ which is a one hop path $q \rightarrow p$ or
- $P_i = (q, q', p)$ which is a two hop path $q \rightarrow q' \rightarrow p$

where q, q', p are distinct nodes and q, q' lie in a single neighborhood $N(a, b+r+1)$, and $q \in N(a,b)$ where, critically, nodes in $N(a,b)$ have committed to the correct message. The existence of these $2t+1$ paths, and the fact that each broadcast neighborhood has at most $t < r/2(2r+1)$ Byzantine faults, is sufficient to prove that reliable broadcast is achieved by the protocol. For simplicity, we can consider $p \in N(a, b+1)$ since the analysis is nearly identical for the cases where $p \in N(a+1, b)$, $p \in N(a-1, b)$, and $p \in N(a, b-1)$.

The node p lies in $N(a,b+1) - N(a,b)$ and can be considered to have location $(a-r+z, b+r+1)$ where $0 \leq z \leq 2r$. Now, summarizing the proof in [6], we demonstrate that there exist $r(2r+1)$ node-disjoint paths $P_1, ..., P_{r(2r+1)}$ all lying within the same neighborhood:

- **One-Hop Paths:** the set of nodes $A_p = \{q(x,y) \mid (a-r) \leq x \leq (a+z)$ and $(b+1) \leq y \leq (b+r)\}$ lie in $N(a,b)$ and are neighbors of p. Therefore, there are $r(r+z+1)$ paths of the form $q \rightarrow p$ where $q \in A_p$.
- **Two-Hop Paths:** consider the sets $B_p = \{q(x,y) \mid (a+z+1) \leq x \leq (a+r)$ and $(b+1) \leq y \leq (b+r)\}$ and $B'_p = \{q'(x',y') \mid (a+z+1-r) \leq x' \leq a$ and $(b+r+1) \leq y' \leq (b+2r)\}$. The nodes in B_p lie in $N(a,b)$ while the nodes in B'_p lie in $N(p)$. Moreover, the set B'_p is obtained by shifting left by r units and up by r units. Therefore, there is a one-to-one mapping between the nodes in B_p and the nodes in B'_p. For $u \in B_p$, we will call the corresponding node $u' \in B'_p$, the *sister node* of u. Note that each node has at most two sister nodes; this can be seen in Figure 3. Hence, there are $r(r-z)$ paths of the form $q \rightarrow q' \rightarrow p$.

Therefore, there are a total of $r(r+z+1) + r(r-z) = r(2r+1)$ disjoint paths all lying in a single neighborhood $N(a, b+r+1)$. Figure 1 illustrates aspects of the discussion above where $a, b = 0$. Now, note that the predecessor set $G_p = A_p \cup B'_p$ is the set of nodes to which p must listen in order to gather information that will allow it to commit to the correct message.

In Section 5.1 and Section 5.2, we explain our protocols for reliable broadcast under both the fail-stop and Byzantine fault models, respectively. Our protocols rely heavily on the results of Bhandari and Vaidya [6] discussed above. In particular, we assume that each node p knows a predecessor set G_p of nodes to which node p should listen for messages. As we have just reviewed, the existence of G_p is shown by the constructive proofs in [6]. Our protocols specify when each node p should listen to nodes in G_p and when each node p should broadcast the message to which it has committed. The set G_p has size $n = r(2r+1)$ and, in executing our protocol for the case where $t < (r/2)(2r+1)$, we assume that at least $n/2$ of the nodes in G_p are correct. Our algorithms for the Bad Santa problem then apply by having a node sample from nodes in G_p in order to listen for a message.

Error Tolerance in the Bad Santa Protocols and in Reliable Broadcast Protocols: Before describing our reliable broadcast protocols, we first address a possible point of confusion: Previous work under the Byzantine fault model assumed $t < n/2$ whereas in this work we are allowing $t \leq n/2$ *in our algorithms for the single-stream Bad Santa problem.* This should not be construed as contradicting the lower bound proved in [8]. In order to perform reliable broadcast $t < n/2$ must indeed hold true and,

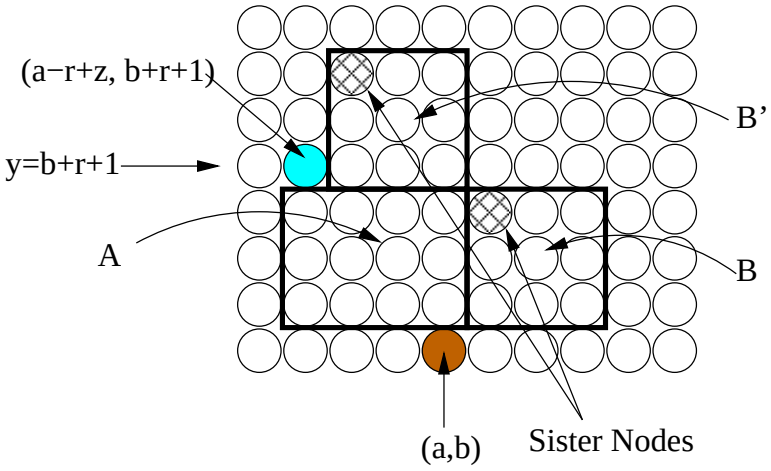

Figure 1: An illustration of the sets A_p, B_p, and B'_p where $z = 0$, $r = 3$ and $a, b = 0$. Node p is located at position $(a - r + z, b + r + 1)$. A pair of sister nodes, one in B' and the other in B, are highlighted.

as we shall see in Section 5.2, this needs to be the case for Stage 1 of our protocol in order to propagate the fingerprint. However, in Stage 2, the set G_p can hold $t \leq n/2$ faulty nodes due to our results on the single-stream Bad Santa problem.

Reliable Broadcast Along a Corridor: The presentation of our protocols is limited to demonstrating reliable broadcast along a corridor of width $2r + 1$ moving along the positive y-coordinates. That is, we show reliable broadcast for a node $p(x, y)$ where $-r \leq x \leq r$ and $y \geq 0$. This greatly simplifies the description of our results. Furthermore, it is easy to see that reliable broadcast is possible along other corridors traversing the x-coordinates or negative y-coordinates using a synchronization of sending and listening similar to what we describe. The grid can be covered piece-wise with such rectilinear corridors in a number of ways; for example, a spiral suffices (see Figure 2). Alternatively, corridors can be appended in many other ways in order to achieve propagation of a message depending on scenario in question. In any event, proving reliable broadcast for this corridor is sufficient to prove reliable broadcast for the grid in general.

5.1 Protocol for Fail-Stop Faults

We describe our reliable broadcast protocol that tolerates fail-stop faults; the proof is deferred until the end of Section 5.2 since it is subsumed by the proof for the Byzantine case. The pseudocode below shows how broadcast can be achieved along a corridor of width $2r + 1$, where $-r \leq x \leq r$, moving along the positive y-coordinates. As mentioned earlier, restricting the movement in this way greatly simplifies our presentation without sacrificing completeness.

We assume that the nodes in the network know the time slot when the source node will broadcast a message. We will let t_{start} denote the time slot at which the source sends out a message m. The source node located at (0,0) broadcasts m at time slot t_{start} and all correct nodes in $N(0, 0)$ are assumed to receive m from the source and commit internally. Node in $N(0, 0)$ then broadcast that they have committed to m for the next $2r$ consecutive rounds during their respective allotted time slots.

We now describe how each node $p(x, y)$, for $-r \leq x \leq r$ and $y \geq r+1$, listens for and sends messages and, finally, how it broadcasts its committal. Let t_q denote the time slot when q is scheduled to broadcast in round $t_{start} + 2(y - r)$. Using t_q values, each node p creates an ordered set $S_p \subset G_p$ where the elements of S_p are chosen according to the $(1, O(\sqrt{n}))$ strategy for the Bad Santa problem. Node p then awakens from the energy-efficient sleep mode and listens (in order) to nodes in S_p in round $t_{start} + 2(y - r)$. If at any point, p receives a message, it commits to this message internally. During the course of the protocol, node p also facilitates the passage of messages along the two-hop paths. While node p has not committed

internally, p listens to each sister node $u(x'', y'')$ in round $t_{start} + 2(y'' - r) + 1$. If p receives a message, then p does the following: (1) commits internally to this message and (2) during its assigned slots p broadcasts m for $2r$ consecutive rounds starting at round $t_{start} + 2(y'' - r) + 2$. Finally, in terms of sending, if at any time a node $p(x, y)$ has committed internally to a message in round $t_{start} + 2(y - r)$ (i.e. used the Bad Santa protocol to commit), p waits until round $t_{start} + 2(y - r) + 1$ and then broadcasts its message for $2r$ consecutive rounds during its assigned time slots. Again, note that in the following pseudocode, each node $p(x, y)$ is such that $-r \leq x \leq r$ and $y \geq 0$.

$(1, O(\sqrt{n})$ Reliable Broadcast for the Fail-Stop Fault Model

1. At time slot t_{start}, the source $d(0, 0)$ does a one-time local broadcast of m and each node in $N(d)$ commits internally to m.

2. All nodes in $N(0, 0)$ broadcast their committal to m for the next consecutive $2r$ rounds.

   The following portion of the protocol is followed by all nodes not in $N(0, 0)$:

3. If node $p(x, y)$ has committed internally to a message in round $t_{start} + 2(y - r)$ (i.e. in Step 5), then p waits until round $t_{start} + 2(y - r) + 1$ and then broadcasts its message for $2r$ consecutive rounds during its assigned time slots.

4. While node $p(x, y)$ has not committed internally to a message, node p listens to each sister node $u(x'', y'')$ in round $t_{start} + 2(y'' - r) + 1$. If p receives the message m from u, then p does the following: (1) commits internally m and (2) during its assigned slots p broadcasts m for $2r$ consecutive rounds starting at round $t_{start} + 2(y'' - r) + 2$.

5. While node $p(x, y)$ has not committed internally to a message, p does the following. For a node $q \in G_p$, let t_q denote the time slot when q is scheduled to broadcast in round $t_{start} + 2(y - r)$. Using t_q values, node p creates an ordered set $S_p \subset G_p$ where the elements of S_p are chosen according to the $(1, \sqrt{n})$ Bad Santa strategy. Then p does the following:

   - Node $p(x, y)$ listens to $q \in S_p$ in round $t_{start} + 2(y - r)$. If at any point p receives a message m, then p commits to m internally, breaks the for-loop and proceeds to Step 3.

5.2 Protocol for Byzantine Faults

Our protocol for the Byzantine fault model runs in two stages. In the first stage, the source propagates a fingerprint $f(m)$ of the message m it wants to broadcast. This fingerprint is assumed to be of size at least $\lg^2 |m|$ bits. Propagation of $f(m)$ is again done using the algorithm in [6]. The second stage is very similar to the previous protocol for the fail-stop faults. In the second stage, the source broadcasts the message m at time slot t_{start} and all correct nodes in $N(0, 0)$ are assumed to receive m from the source and commit internally. Each node $q(x', y') \in N(0, 0)$ then broadcasts its committal to m over the next $2r$ consecutive rounds. A node p listens to messages from a set G_p just as in the protocol for fail-stop model. The difference occurs when, at any point a message m' is received. Node p then checks $f(m')$ against the fingerprint f_{maj} to which it committed in the first stage. If they match, p commits to m' internally and executes the broadcast instructions mentioned previously.

We assume that the network alternates between the first stage, where nodes are constantly awake, and the second stage, where nodes are achieving significant power savings. For instance, it is plausible that internal software could synchronize periodic change-overs between these two stages in much the same way that radio network alternate periodically between sleep and fully active states to conserve power in practice. These details are outside the scope of our work and we do not discuss them further.

Finally, note that a faulty node might broadcast an incorrect message m' such that $|m'| > |m|$ where m is the correct message. To avoid complications, we assume that nodes in the network know the size of m and, therefore, can stop listening after receiving $|m|$ bits. For instance, this could be implemented

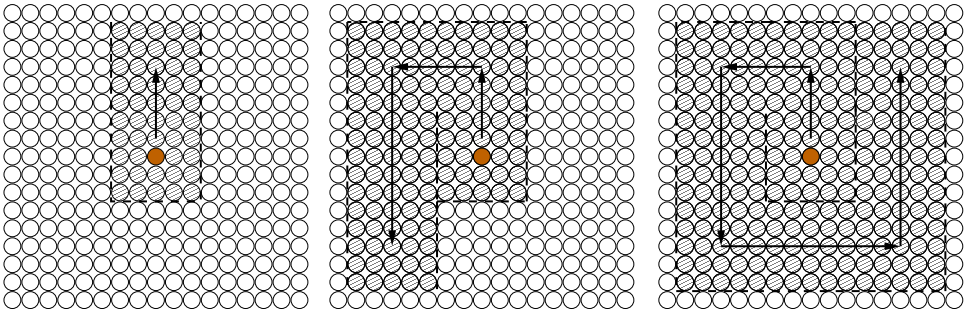

Figure 2: The source is denoted by the brown node which has location $(0,0)$. (A) Movement in the positive y direction establishing a $(2r+1)\times(2r+1)$ square of committed nodes above $N(0,0)$; this is the corridor we explicitly address in the proofs of correctness for our protocols. (B) Spiraling out from $N(0,0)$, movement in the negative x direction and then the negative y direction. (C) Further depiction of the spiral expansion of committed nodes along a corridor of width $2r+1$.

by having the source broadcast the message size in the first stage or having a predefined upper limit on messages size. The details of such solutions would be dictated by context and we omit further discussion of this issue.

$(1, O(\sqrt{n}))$ **Reliable Broadcast for the Byzantine Fault Model**

Stage 1:

1. At time t_0, the source uses the reliable broadcast protocol of [6] to broadcast the fingerprint $f(m)$ to all nodes in the grid.

Stage 2:

2. At time slot t_{start}, the source $d(0,0)$ does a one-time local broadcast of m and each node in $N(d)$ commits internally to m.

3. All nodes in $N(0,0)$ broadcast their committal to m for the next consecutive $2r$ rounds.

The following portion of the protocol is followed by all nodes not in $N(0,0)$:

4. If node $p(x,y)$ has committed internally to a message in round $t_{start}+2(y-r)$ (i.e. in Step 6), p waits until round $t_{start}+2(y-r)+1$ and then broadcasts its message for $2r$ consecutive rounds during its assigned time slots.

5. While node $p(x,y)$ has not committed internally to a message, node p listens to each sister node $u(x'',y'')$ in round $t_{start}+2(y''-r)+1$. If the message m_u that p receives from u equals the f_{maj} value, then p does the following: (1) commits internally m_u and (2) during its assigned slots p broadcasts m_u for $2r$ consecutive rounds starting at round $t_{start}+2(y''-r)+2$.

6. While node $p(x,y)$ has not committed internally to a message, p does the following. For a node $q \in G_p$, let t_q denote the time slot when q is scheduled to broadcast in round $t_{start}+2(y-r)$. Using t_q values, node p creates an ordered set $S_p \subset G_p$ where the elements of S_p are chosen according to the $(1,\sqrt{n})$ Bad Santa strategy. Then p does the following:

   - Node $p(x,y)$ listens to $q \in S_p$ in round $t_{start}+2(y-r)$. In listening to each q, p will obtain a value m_q (or nothing, if q is Byzantine and sends nothing). If at any point $f(m_q)$ equals the f_{maj} value of p, then p commits to m_q internally, breaks the for-loop and proceeds to Step 4.

We now establish the following preliminary lemma which we will need for our protocols. Label the set of nodes in the corridor as $S_{cor} = S_{x,cor} \cup S_{y,cor}$ where $S_{x,cor} = \{q(x', y') \mid (r+1 \leq x' \leq x) \wedge (y - r \leq y' \leq y + r)\}$ and $S_{y,cor} = \{q(x', y') \mid (-r \leq x' \leq r) \wedge (0 \leq y' \leq y + 2r)\}$. Figure 3(a) illustrates a corridor for $r = 3$. Finally, recall that a *round* is one iteration through the broadcast schedule.

The following Lemma 7 is useful for our Byzantine-tolerant protocols. In particular, it provides an analysis of the previous protocol in [6] with the minor modification that a node waits for the (at most) two messages from its sister nodes before issuing HEARD messages. We then later use this result to address the necessary delay between Stage 1, where a fingerprint is propagated, and Stage 2 of our protocol, when the full message is sent. Note that in Lemma 7, we deal with arbitrary x and y values.

Lemma 7. *Assume a broadcast schedule where no collisions occur and each node can broadcast once every round as discussed in Section 1.2.2. Consider a source, $d(0,0)$, that broadcasts a fingerprint f at time slot t_0 under either the fail-stop or Byzantine fault models where $t < \frac{r}{2}(2r+1)$. Then by using the protocol of [6], node $p(x,y)$ is able to commit to f by round $t_0 + 2(|x| + |y|)$.*

Proof. We are essentially following the argument for correctness given in [6] and discussed in Section 5; however, we are restricting our view to those nodes in S_{cor}. That is, nodes in S_{cor} will only accept messages from other nodes in S_{cor} and they will ignore all messages they receive from nodes outside the corridor. Clearly, this can only result in a slowdown in the propagation of the broadcast value; moreover, the rectilinear shape of the corridor can only slow down the rate of propagation in comparison to the original propagation described in [6]. An argument identical to that in [6] can be used to show that each correct node $q(x', y') \in S_{cor}$ will commit to the correct fingerprint by receiving messages along at least $2t+1$ node disjoint paths of the form (u_i, q) and (u_i, u_i', q) as shown in Figure 3(a). While we do not repeat the entire argument here, Figure 3(b) illustrates the set G_p for each node p in a row of the corridor along increasing y-values. That is, the regions A_p, B_p and B_p' are illustrated for each position in the context of the proof discussed in Section 5.

We now consider the time required until $p(x,y)$ can commit to f regardless of which nodes in the corridor fail; p does so by listening to the nodes in G_p. Without loss of generality, assume that x, y are positive coordinates and that the broadcast first moves nodes in $S_{y,cor}$ (moving up) and then along nodes in $S_{x,cor}$ (moving right). At t_0, the source broadcasts f and all nodes in $N(0,0)$ commit to f. Consider a node $q(a, r+1)$ where $-r \leq a \leq r$. It takes at most one round for q to receive messages along paths of the form (u_i, q) from region A. Concurrently, in this one round, nodes u_i can transmit messages to nodes u_i' along paths of the form (u_i, u_i', q) (region B to B') where the HEARD messages from the (at most) two sister nodes are appended in a single message. At most an additional round is required to send from nodes u_i to q. Therefore, at most two rounds are required before q can commit. Note that this holds for all nodes with coordinates $(a, r+1)$ for $-r \leq a \leq r$; this entire row can commit after at most two rounds. It follows that all nodes up to and including row y in $S_{y,cor}$ are committed to f after $t_0 + 2(y + r)$ rounds; the remaining r rows in $S_{y,cor}$ do not commit. An identical argument shows that all nodes in $S_{x,cor}$ are committed to f after $t_0 + 2(x - r)$ rounds. Therefore, p commits after at most $t_0 + 2(x + y)$ rounds; if x and y can take on negative values, this becomes $t_0 + 2(|x| + |y|)$. □

The next lemma proves that, if we assume that the adversary cannot cause a collision with f_{maj}, then each node can commit to the correct message using our protocol. In particular, it establishes that the second stage of our protocol achieves the $2t+1$ connectedness necessary for reliable broadcast. The lemma also establishes that the broadcasting and receiving actions by each node are correct. Finally, the resource costs per node for Stage 2 follow immediately. Note that this stops short of proving Theorem 4 since the issue of fingerprints has not yet been addressed. While we include it for completeness, we stress that the $2t+1$ connectedness component of the proof is essentially an adaptation of the proof found in [6] which was reviewed in the beginning of Section 5. Again, the proof focuses on movement along the positive y-coordinates along a corridor of width $2r+1$ where $-r \leq x \leq r$. Figure 4 illustrates how our protocol proceeds when $r = 3$.

Lemma 8. *Assume a broadcast schedule where no collisions occur and each node can broadcast once every round as discussed in Section 1.2.2. Furthermore, assume each node already possesses f_{maj} prior*

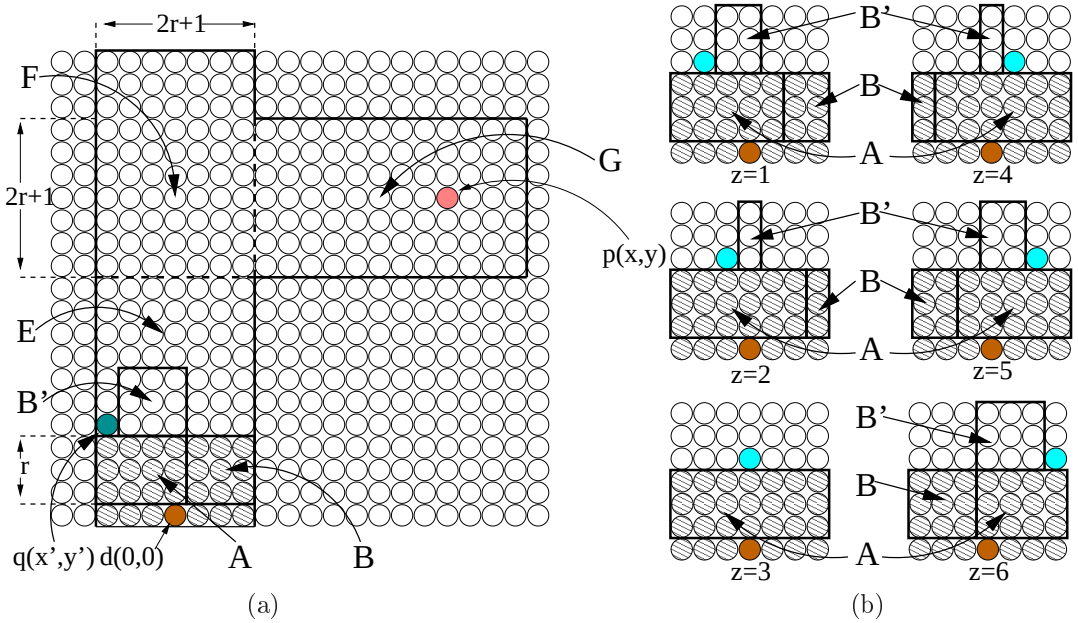

Figure 3: Let p be a node that is not at the boundary of width r in the grid. (a) A depiction of a corridor for $r = 3$. Together the nodes in region E and F constitute $S_{y,cor}$ while the nodes in G constitute $S_{x,cor}$. Node disjoint paths of the form (u_i, q) originate from nodes u_i in region A. As discussed in [5, 6], node disjoint paths of the form (u_i, u'_i, q) originate from nodes u_i in region B and traverse through nodes u'_i in B' to reach node q. (b) The regions A, B and B' are illustrated for each node along a row of $S_{y,cor}$. The value for z is given for each position in the context of the proof reviewed in Section 5.

to receiving any other messages and that if a message m received by a correct node p corresponds to the fingerprint f_{maj} propagated in the first step, then m is the correct message. Under these assumptions, the $(1, O(\sqrt{n}))$ Reliable Broadcast for the Byzantine Fault Model protocol has the following properties:

- *Each node $p(x, y)$ where $-r \leq x \leq r$ (except those on the boundary of width r if the grid is finite) commits to the correct message m by round $\max\{2(y - r), 0\}$.*
- *Each node is awake for $O(\sqrt{n})$ time slots in expectation. Each node sends and receives $O(\sqrt{n}|m|)$ bits in expectation.*

Proof. For simplicity, we normalize such that t_{start} is round 0. Our proof is by induction and throughout we assume that each node has an x-coordinate such that $-r \leq x \leq r$:

Base Case: Each node in $N(0, 0)$ commits to the correct message m immediately upon hearing it directly from the dealer. Therefore, each node $p(x, y) \in N(0, 0)$ commits to m by round $0 \leq \max\{2(y - r), 0\}$.

Induction Hypothesis: For simplicity, we will assume as before that $p \in N(a, b + 1)$ where $-r \leq a \leq r$; the other cases for proving the statement for $p \in PN(a, b)$ follow by symmetry. In this context, the induction hypothesis is as follows: if each $p'(x', y') \in N(a, b)$ has committed to m by round $2(y' - r)$, then each correct node $p(x, y) \in N(a, b + 1) - N(a, b)$ is able to commit to m by round $2(y - r)$.

Induction Step: As we reviewed before in the beginning of Section 5, we show $2t + 1$ connectedness in a single neighborhood. We will argue simultaneously about the time required for p to hear messages along these disjoint paths. The node $p(x, y)$ lies in $N(a, b + 1) - N(a, b)$ and can be considered to have location $(a - r + z, b + r + 1)$ where $0 \leq z \leq r$ (the case for $r + 1 \leq z \leq 2r$ follows by symmetry). We demonstrate that there exist $r(2r + 1)$ node-disjoint paths $P_1, ..., P_{r(2r+1)}$ all lying within the same neighborhood and that the synchronization prescribed by our protocol is correct:

- **One-Hop Paths:** the set of nodes $A_p = \{q(x,y) \mid (a-r) \leq x \leq (a+z) \text{ and } (b+1) \leq y \leq (b+r)\}$ lie in $N(a,b)$ and are neighbors of p. Therefore, there are $r(r+z+1)$ paths of the form $q \rightarrow p$ where $q \in A_p$.

  By their position relative to $p(x,y)$, each correct node $q(x',y') \in A_p$ is such that $y-r \leq y' \leq y-1$. Therefore, by the induction hypothesis, a correct node $q \in A_p$ commits in round $2(y-2r)$ at the earliest and $2(y-r-1)$ at the latest. Consequently, a correct node in A_p starts broadcasting its committals in round $2(y-2r)+1$ at the earliest and $2(y-r-1)+1$ at the latest. In the former case, recall that broadcasting occurs for $2r$ rounds, which means that q is broadcasting from round $2(y-2r)+1$ to $2(y-r)$, inclusive, at the earliest. In the latter case, q is broadcasting from $2(y-r-1)+1$ to $2(y-1)$, inclusive. Therefore, all correct nodes in A_p are broadcasting a committal message in round $2(y-r)$ and so $p(x,y)$ can receive a message from each correct node in A_p in this round.

- **Two-Hop Paths:** consider the sets $B_p = \{q(x,y) \mid (a+z+1) \leq x \leq (a+r) \text{ and } (b+1) \leq y \leq (b+r)\}$ and $B'_p = \{q'(x,y) \mid (a+z+1-r) \leq x \leq (a) \text{ and } (b+r+1) \leq y \leq (b+2r)\}$. The nodes in B_p lie in $N(a,b)$ while the nodes in B'_p lie in $N(p)$. Moreover, the set B'_p is obtained by shifting left by r units and up by r units. Recall that there is a one-to-one mapping between the nodes in B_p and the nodes in B'_p; these are sister nodes. There are $r(r-z)$ paths of the form $q \rightarrow q' \rightarrow p$.

  Consider a correct node $q(x',y') \in B_p$ and its sister node $q'(x'',y'') \in B'_p$. Again, given the location of $q(x',y')$ relative to $p(x,y)$, by the induction hypothesis, the earliest $q \in N(a,b)$ has committed is $2(y-2r)$ and the latest is $2(y-r-1)$. Therefore, by protocol, q starts broadcasting its committal $2r$ times starting in round $2(y-2r)+1$ at the earliest and $2(y-r-1)+1$ at the latest. The sister node of q, $q' \in B'_p$, listens to q in the first round that q broadcasts. If q' receives a correct m, then q' broadcasts this $2r$ times; therefore, this occurs in round $2(y-2r)+2 = 2(y-2r+1)$ at the earliest and $2(y-r-1)+2 = 2(y-r)$ at the latest. In the former case, recall that q' broadcasts for $2r$ consecutive rounds and therefore is broadcasting until round $2(y-r+1)-1 > 2(y-r)$. Therefore, all correct nodes in B'_p with a message to broadcast are doing so in round $2(y-r)$ and so $p(x,y)$ can hear a message from any such $q' \in B'_p$ in this round.

Therefore, there are a total of $r(r+z+1)+r(r-z) = r(2r+1)$ node-disjoint paths from $N(a,b)$ to $PN(a,b)$, all lying in in a single neighborhood $N(a,b+r+1)$. By our argument above, each correct node $p(x,y)$ receives the one-hop and two-hop messages over these paths by round $2(y-r)$. We note that (1) more than half of these paths will provide the correct message and (2) the sampling follows the Bad Santa protocol which is a Las Vegas algorithm. Therefore, we are guaranteed that p will obtain a message m that corresponds to f_{maj}. Finally, by our initial assumption regarding the inability of the adversary to forge a collision, this means that m is the correct message.

We now analyze the resource bounds for our protocol. Consider the situations where p must deal with (either broadcasting or receiving) a message: (1) p receives messages in order to commit, (2) p broadcasts it has committed, and (3) p facilitates two-hop messages. We consider each case. To address (1), note that p uses the Bad Santa protocol; while in the streaming problem, we attempt to obtain a 1 at unit cost per query, here node p is attempting to select a correct node at the cost of listening to $|m|$ bits per selection.[6] This method of sampling from G_p means p receives $O(\sqrt{n})$ messages in expectation. To address (2), note that p broadcasts that it has committed $2r = O(\sqrt{n})$ times. To address (3), we consider $p \in PN(a,b+1)$ as before, and note that p belongs to many B'_q sets for different nodes q; however, regardless of which B'_q set, p only ever has two sister nodes. Therefore, considering broadcast along the x and y coordinates, the number of sister nodes is $O(1)$; the number of broadcasts due to two-hop paths is thus $O(r)$. In conclusion (not counting the fingerprint, since we are dealing only with the second stage of our protocol) each node is awake for $O(\sqrt{n})$ time slots in expectation, sends $O(\sqrt{n}|m|)$ bits and receives $O(\sqrt{n}|m|)$ bits. □

With Lemma 7 and Lemma 8 in hand, we can now give the proof for Theorem 4:

[6] Selecting a random node is necessary; if not, the adversary might have faulty nodes send correct fingerprints in the first round and, if p selects nodes from G_p in a deterministic fashion, the adversary may force p to listen to many messages that do not hash to f_{maj}.

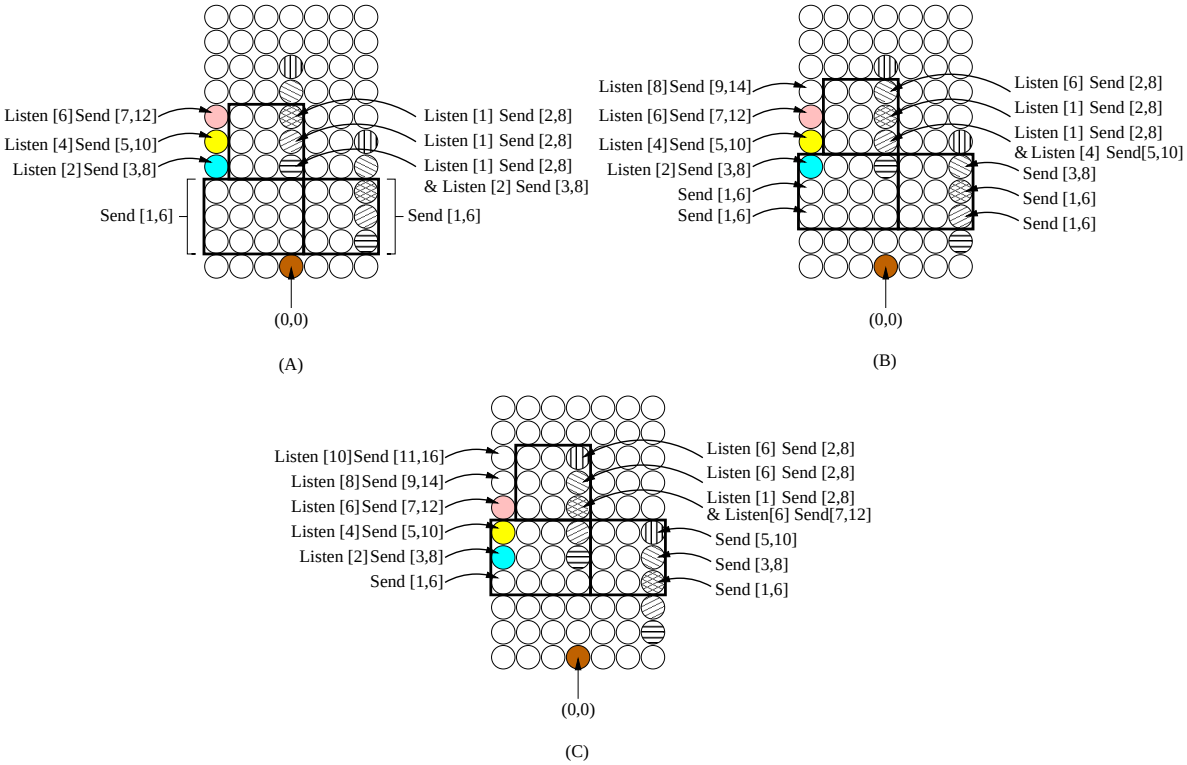

Figure 4: A depiction of the $(1, O(\sqrt{n}))$ Reliable Broadcast for the Byzantine Fault Model protocol for $r = 3$. Times for broadcasting and receiving are denoted by $[a, b]$ which denotes rounds a through b inclusive. (A) Shows how a node p in row 4 can commit by listening to nodes in $G_p = A_p \cup B'_p$; we focus on nodes on the left-most edge. Note the node in row 4 marked with horizontal lines. This node acts as part of B'_p while also committing as other nodes in row 4 do; later, it will act as a node in B'_i and A_j for other nodes i and j by sending in rounds $[3, 8]$. Note that this node is sending from round 2 to 8 (inclusive) but we separate this into $[2, 8]$ and $[3, 8]$ to make the different roles explicit. (B) & (C) Depictions of the timing of broadcasting and receiving as nodes in rows 5 and 6 commit, respectively.

Proof. We begin by proving correctness and we start with Stage 1. Stage 1 of the protocol is no different than the broadcast presented in [6] where the value being transmitted is a fingerprint. Consequently, every correct node will be able to derive a majority fingerprint f_{maj}.

We now analyze Stage 2. Lemma 8 assumes that (1) we have an appropriate schedule, (2) each node has f_{maj} prior to receiving any other messages, and (3) if m corresponds to f_{maj} then m is correct. We go about addressing these three criteria:

- First, we can assume the schedule of [8] which satisfies the properties required by Lemma 8.
- Second, consider all nodes in a square of size $3(2r + 1) \times 3(2r + 1)$ centered about $(0, 0)$; the node at the top-right has position $(3r + 1, 3r + 1)$. By selecting t_{start} at least $2(6r + 2)$ rounds after the time of sending of the fingerprint, t_0 (that is $t_{start} \geq t_0 + 2(6r + 2)$), then Lemma 7 guarantees that by time t_{start}, all nodes in a square of size $3(2r + 1) \times 3(2r + 1)$ centered about $(0, 0)$ will have committed to the fingerprint. If we assume, as mentioned earlier prior to presenting the pseudocode, that the message expands via a spiral corridor of width $2r+1$ from $N(0, 0)$, then this guarantees that the propagation of the fingerprint will always be sufficiently far ahead of the propagation of the full message to allow nodes to first commit to the fingerprint. Note that if m is propagated in a different fashion (i.e. not a spiral) then the timing offset would need to be adjusted accordingly.
- Third, by assumption, f is a secure hash function and the size of the fingerprint is $\lg^2 m$. Therefore,

given $f(x)$, the probability that the adversary obtains a value x' such that $f(x') = f(x)$ is $2^{-\lg^2 |m|} = |m|^{-\lg |m|}$. It will take the adversary superpolynomial time in m to forge such an x' and so f_{maj} will correspond to the correct value m. Recall that s is the number of computational steps afforded to the adversary; i.e. the number of times the adversary can create an input x', apply f to x' and check for a match between the output fingerprint $f(x')$ and f_{maj}. Therefore, given that p receives a message from a correct node in G_p that when hashed matches the fingerprint to which p committed, with error $O(s/|m|^{\lg |m|})$, this message is the correct message sent by the source where s is the number of computational steps available to the adversary.

Finally, we analyze resource costs. Lemma 8 confirms the amount of awake time specified by Theorem 4 after the sending of the fingerprint. Regarding the bit complexity over both Stages 1 and 2, we need consider the additional cost due to sending the fingerprint. Each node p broadcasts and receives $O(r^2) = O(n)$ fingerprints, for a total of $O(n \log^2 |m|)$ bits in Stage 1. Therefore, the expected bit complexity over both stages is $O(n \log^2 |m| + \sqrt{n}|m|)$. □

The above proof essentially subsumes the proof for Theorem 3; however, we include it here for completeness:

Proof. The proof of correctness for the fail-stop model differs in two places from the proof of Theorem 4. For criteria (2), there is no need for a fingerprint. For criteria (3), since messages are never corrupted, only lost if a fault occurs, p is guaranteed that the message it receives from a correct $q \in G_p$ is correct. Finally, for the fail-stop model, the resource costs are easy to analyze. The awake times due to listening follow directly from the fact that each node broadcasts $O(\sqrt{n})$ times and uses the Bad Santa problem for listening; therefore, a total of $O(\sqrt{n})$ time slots in expectation. In terms of bit complexity, each node p broadcasts $|m|$ for r rounds times and listens to m once. Therefore, p broadcasts $\Omega(\sqrt{n}|m|)$ bits and receives $|m|$ bits. □

It may seem that, with some modifications to the protocol, we can employ a multi-stream Bad Santa strategy to achieve further expected savings. We now explain why this is not the case. Note that such a change would require each node to send $O(r \cdot k \cdot |m|)$ bits while reducing the expected listening cost to $O(k \cdot |m|)$. However, since the costs for sending and receiving are of the same magnitude, we do not achieve an overall asymptotic savings when we consider the addition of these two communication costs.

Finally, we comment on the difference in running time between our algorithms for the fail-stop and Byzantine fault models. Clearly, the need to propagate a fingerprint in the Byzantine case incurs additional time. However, as we have seen, the dealer need wait only $2(6r + 2)$ rounds after sending the fingerprint before broadcasting the full message.

5.3 Reliable Broadcast when $t \leq (1 - \epsilon)\frac{r}{2}(2r + 1)$

When $t \leq (1 - \epsilon)\frac{r}{2}(2r + 1)$ for any constant $\epsilon > 0$, we show how to achieve an even larger energy savings by employing the $(k + 1, O(\log^{(k)}(n/2) + k))$ strategy to the Bad Santa problem for $k \geq 1$. As we will show, a correct node may listen to at least $(r/2)(2r + 1)$ messages, of which a $(1 - \epsilon)$-fraction may be faulty; therefore, we are now allowing more than a $1/2$ fraction of paths to deliver faulty messages. We know by Theorem 8, that employing a single stream Bad Santa strategy in this scenario does not yield any asymptotic savings; hence, we deal only with $k \geq 1$. We also point out that, in actuality, our results hold for $t \leq (1 - \epsilon)(1 + r + r^2)$ which is larger than $(1 - \epsilon)(r/2)(2r + 1)$ by an amount of $(1 - \epsilon)(1 + r/2)$. However, asymptotically, this difference is negligible and we phrase the result in this manner to illustrate that we are within an arbitrary constant fraction of the optimal tolerance.

In this case, we present a Byzantine fault-tolerant reliable broadcast protocol; the protocol for tolerating fail-stop faults is straightforward and we omit it. The protocol is very similar to our $(1, O(\sqrt{n}))$ Reliable Broadcast for the Byzantine Fault Model presented in Section 5.2 where here we use the $(k + 1, O(\log^{(k)}(n) + k))$ strategy; however, there are important distinctions. In particular, the sets A_p, B'_p and B_p are defined in a slightly different manner in the correctness proof for our protocol later on. Furthermore, each node broadcasts for $O(k)$ (rather than r) consecutive rounds and the synchronization of broadcasting and receiving is altered. Essentially, r rows of a corridor are committing every $k+2$ rounds; this is different from the previous protocols where each row committed in a different round. The B'_p and B_p sets are

no longer changing in size with the position of the node p; rather, these sets are $r \times r$ squares. The pseudocode is given below and, again, deals with movement along the positive y-coordinates in a corridor of width $2r+1$, where nodes have an x-coordinate such that $-r \leq x \leq r$.

$(k+1, O(\log^{(k)}(n)+k))$ **Reliable Broadcast for the Byzantine Fault Model**

Stage 1:

1. At time slot t_0, the source uses the reliable broadcast protocol of [6] to broadcast the fingerprint $f(m)$ to all nodes in the grid.

Stage 2:

2. At time slot t_{start}, the source $d(0,0)$ does a one-time local broadcast of m and each node in $N(d)$ commits internally to m.
3. All nodes in $N(0,0)$ broadcast their committal to m for the next consecutive $k+2$ rounds.

The following portion of the protocol is followed by all nodes not in $N(0,0)$:

4. If node $p(x,y)$ has committed internally to a message via listening to a set $S_{p,i}$ for $i = 0, ..., k$ (i.e. Step 6), node p uses its allotted time slot to broadcast this fact for $k+2$ consecutive rounds; that is, from round $(k+2)\left(\lfloor \frac{y-1}{r} \rfloor\right)+1$ to $(k+2)\left(\lfloor \frac{y-1}{r} \rfloor + 1\right)$ inclusive.
5. While node $p(x,y)$ has not committed to a message, node p listens to each sister nodes in round $(k+2)\left(\lfloor \frac{y-r-1}{r} \rfloor\right)+1$. If the message m_u that p receives from u equals the f_{maj} value, then p does the following: (1) commits internally m_u and (2) during its assigned slots p broadcasts m_u for $k+1$ consecutive rounds: from round $(k+2)\left(\lfloor \frac{y-r-1}{r} \rfloor\right)+2$ to round $(k+2)\left(\lfloor \frac{y-r-1}{r} \rfloor + 1\right)$, inclusive.
6. While node $p(x,y)$ has not committed internally to a message, p does the following. For a node $q \in G_p$, let t_q denote the time slot when q is scheduled to broadcast in round $(k+2)\left(\lfloor \frac{y-r-1}{r} \rfloor\right)+2$. Using t_q values, node p creates ordered sets $S_{p,0}, ..., S_{p,k}$ where $S_{p,i} \subset G_p$ for $i = 0, ..., k$ where the elements of each $S_{p,i}$ are chosen according to the $(k+1, O(\log^{(k)}(n)+k))$ Bad Santa strategy. Then for $i = 0, ..., k$, p does the following:
   - Node $p(x,y)$ listens to each node $q \in S_{p,i}$ for $k+1$ consecutive rounds: that is, from round $(k+2)\left(\lfloor \frac{y-r-1}{r} \rfloor\right)+2$ to round $(k+2)\left(\lfloor \frac{y-r-1}{r} \rfloor + 1\right)$. If at any point q receives a message m_q such that $f(m_q)$ equals the f_{maj} value of p, then p commits to m_q internally, breaks the for-loop and proceeds to Step 4.

To avoid possible confusion, we draw attention to the fact that nodes acting as members of A_p sets broadcast for $k+2$ times, even though the corresponding Bad Santa protocol uses $k+1$ streams. This is because of the extra delay of one round incurred by the two-hop messages; we note that nodes in B'_p sets that facilitate these messages broadcast for $k+1$ consecutive rounds. The correctness of this protocol can be demonstrated in a similar fashion to the preceding protocols; however, there is a difference in that now the proof deals with all nodes in r rows rather than a single row. Figure 5 illustrates how this protocol proceeds when $r = 3$ and $k = 3$. For completeness, establish Theorem 6 although we again only consider movement along the positive y-coordinates.

Proof. The proof is again by induction and, for simplicity, we assume $t_{start} = 0$ and we again assume that each node in the corridor has an x-coordinate such that $-r \leq x \leq r$. We show $2t+1$ connectedness inside a single neighborhood and that each node $p(x,y)$, for $-r \leq x \leq r$, commits by round $(k+2)\left(\max\left\{\lfloor \frac{y-r-1}{r} \rfloor + 1, 0\right\}\right)$.

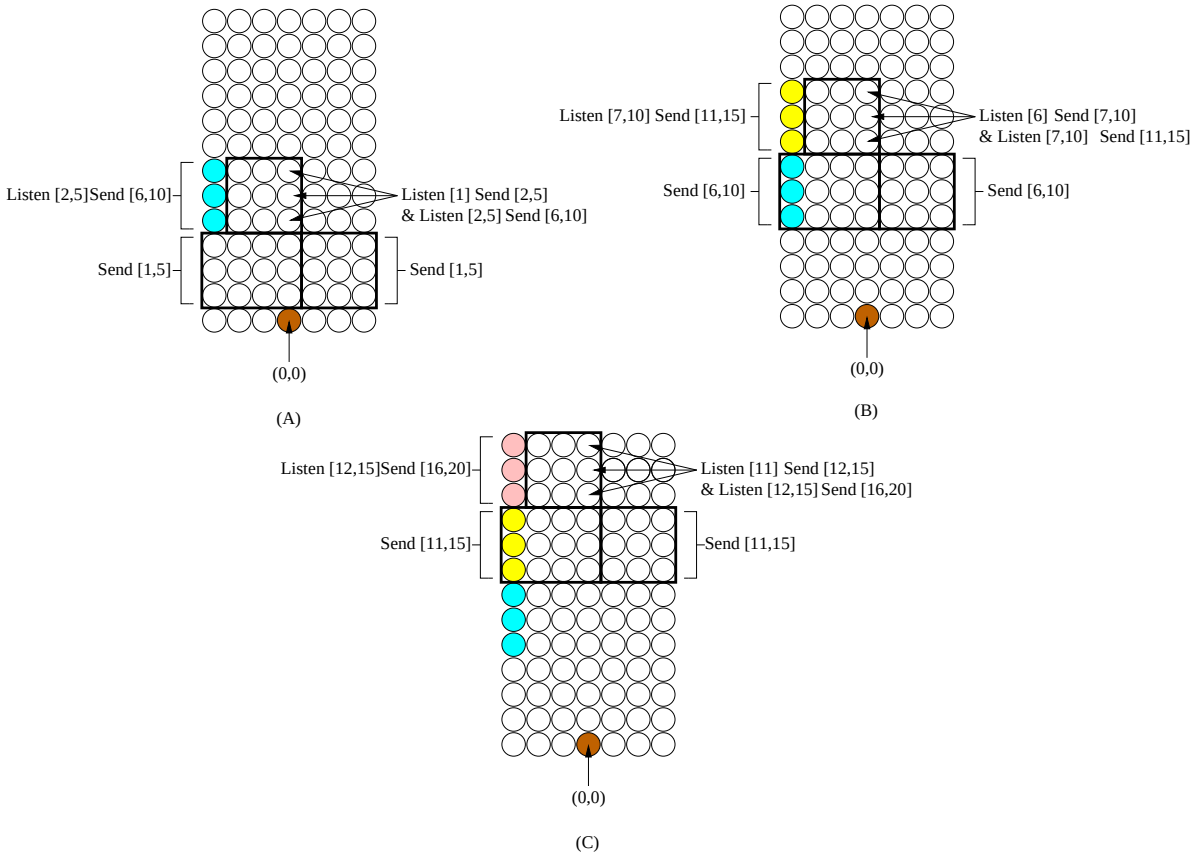

Figure 5: A depiction of the $(k+1, O(\log^{(k)}(n)+k))$ Reliable Broadcast for the Byzantine Fault Model protocol for $r=3$ and $k=3$. Times for broadcasting and receiving are denoted by $[a,b]$ which denotes rounds a through b inclusive. (A) Shows how all nodes in rows 4, 5, and 6 commit; we focus on nodes along the leftmost edge. Node p_1 along the edge in row 4 can commit by listening to all nodes in $G_p = A_p \cup B'_p$. In contrast, p_3 in row 6 can sample from only the top row in A_p, which consists of r nodes, and all of B'_p, which consists of r^2 nodes. (B) & (C) Depictions of the timing of broadcasting and receiving as nodes in rows $7, 8, 9$ and, subsequently, nodes in rows $10, 11, 12$ commit, respectively.

Base Case: Each node in $N(0,0)$ commits to the correct message m immediately upon hearing it directly from the dealer; that is, by round 0.

Induction Hypothesis: Rather than dealing with nodes in $p \in N(a,b+1) - N(a,b)$, our proof differs in that we address all nodes in in $p \in N(a,b+r) - N(a,b)$ i.e. all nodes in the r rows above row b and for simplicity we will assume $b > 0$ (we do not deal with the nodes in $N(0,0)$) and that $-r \leq a \leq r$. In particular, the induction hypothesis is as follows: if each node $p'(x',y')$ in rows $b+1, ..., b+r$ of $N(a,b)$ commit to m in round $(k+2)\left(\left\lfloor \frac{y'-r-1}{r} \right\rfloor + 1\right)$, then each correct node $p(x,y) \in N(a,b+r) - N(a,b)$ commits to m in round $(k+2)\left(\left\lfloor \frac{y-r-1}{r} \right\rfloor + 1\right)$.

Induction Step: We show $2t+1$ connectedness and simultaneously prove the correctness of the timing for broadcasting and receiving. The node $p(x,y)$ lies in $N(a,b+r) - N(a,b)$ and can be considered to have location $(a-r+z, b+r+1+c)$ where $0 \leq z \leq r$ (the case for $r+1 \leq z \leq 2r$ follows by symmetry) and $0 \leq c \leq r-1$. We demonstrate that there exist at least $1+r+r^2$ node-disjoint paths $P_1, ..., P_{1+r+r^2}$ all lying within the same neighborhood and that the synchronization prescribed by our protocol is correct. As we mentioned previously, the sets A_p, B'_p and B_p are defined slightly differently than previously; there are defined below in our proof and Figure 6 depicts these sets.

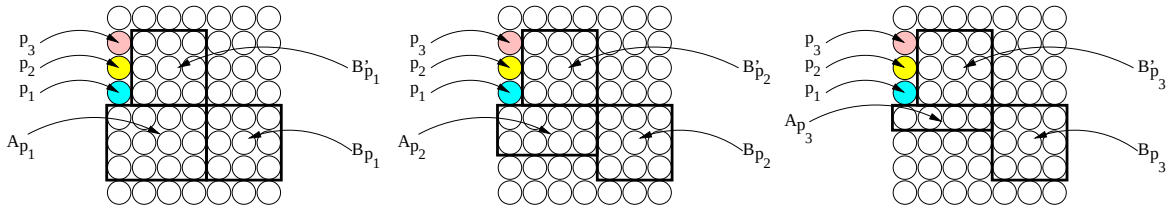

Figure 6: The sets A_{p_i}, B'_{p_i} and B_{p_i} for $i = 1, 2, 3$ and $r = 3$. (A) For the node p_1 with a y-coordinate such that $y \mod r = 1$, the sets are defined the same way. (B) Node p_2 has an A_{p_2} set which consists only of the two top rows of A_{p_1}. (C) Node p_3 has an A_{p_3} set which consists only of the top row of A_{p_1}. Finally, note that $B'_{p_1} = B'_{p_2} = B'_{p_3}$ since p_1, p_2 and p_3 share the same x-coordinate.

- **One-Hop Paths:** the set of nodes $A_p = \{q(x, y) \mid (a - r) \leq x \leq a \text{ and } (b + 1 + c) \leq y \leq (b + r)\}$ lie in $N(a, b + r + 1)$. Therefore, node $p(a - r + z, b + r + 1 + c)$ can receive broadcasts from nodes in at least $r - c \geq 1$ row(s) of A_p which amounts to at least $r + 1$ nodes.

  Consider a correct node $q(x', y')$ in the r rows $b + 1, ..., b + r$ of $N(a, b)$ and recall $p(x, y)$ is in some row $b + r + 1, ..., b + 2r$. The induction hypothesis guarantees that q has committed by round $(k + 2)(\lfloor \frac{y'-r-1}{r} \rfloor + 1) = (k + 2)(\lfloor \frac{y-r-1}{r} \rfloor)$. Then, by the protocol, q broadcasts for $k + 2$ consecutive rounds; that is, from round $(k + 2)(\lfloor \frac{y-r-1}{r} \rfloor) + 1$ to round $(k + 2)(\lfloor \frac{y-r-1}{r} \rfloor) + k + 2$, inclusive. Node p is scheduled to begin listening in round $(k + 2)(\lfloor \frac{y-r-1}{r} \rfloor) + 2$ and so p can receive a message from each such q for $k + 1$ consecutive rounds. Therefore, p hears all one-hop messages by round $(k + 2)(\lfloor \frac{y-r-1}{r} \rfloor + 1)$

- **Two-Hop Paths:** consider the sets $B_p = \{q(x, y) \mid (a + 1) \leq x \leq (a + r) \text{ and } (b + 1) \leq y \leq (b + r)\}$ and $B'_p = \{q'(x, y) \mid (a - r + z + 1) \leq x \leq (a + z) \text{ and } (b + r + 1 - c) \leq y \leq (b + 2r)\}$. The nodes in B_p form an $r \times r$ square within $N(a, b)$ while the nodes in B'_p, again an $r \times r$ square, both lie in the neighborhood $N(a, b + r + 1)$. Note that now the set B'_p is no longer necessarily obtained by shifting left by r units and up by r units; now it is obtained by shifting left by $r - z$ units and up by r units. There is still a one-to-one mapping between the nodes in B_p and the nodes in B'_p; these are sister nodes. There are r^2 paths of the form $q \to q' \to p$.

  From the point of view of $p(x, y)$, consider a correct node $q(x', y') \in B_p$. By the induction hypothesis, q in one of the rows $b + 1, ..., b + r$ of $N(a, b)$ has committed by round $(k + 2)(\lfloor \frac{y'-r-1}{r} \rfloor + 1) = (k + 2)(\lfloor \frac{y-r-1}{r} \rfloor)$. By the protocol, its sister node $q' \in B'_p$ listened at the first of these time slots; hence, q' can receive a message from q in round $(k+2)(\lfloor \frac{y-r-1}{r} \rfloor)+1$. If q' received a correct m, then q' would broadcast m starting in round $(k+2)(\lfloor \frac{y-r-1}{r} \rfloor)+2$ for $k+1$ consecutive rounds. Therefore, q broadcasts a correct message from round $(k+2)(\lfloor \frac{y-r-1}{r} \rfloor)+2$ to round $(k+2)(\lfloor \frac{y-r-1}{r} \rfloor)+k+2 = (k + 2)(\lfloor \frac{y-r-1}{r} \rfloor + 1)$ (inclusive). Node p is scheduled to begin round $(k + 2)(\lfloor \frac{r-r-1}{r} \rfloor + 1)$ and so p can receive a message from each such $q' \in B'_p$ for $k + 1$ consecutive rounds. Therefore, p hears all two-hop messages by round $(k + 2)(\lfloor \frac{y-r-1}{r} \rfloor + 1)$

We have shown that there are at least $1 + r + r^2$ node-disjoint paths from $N(a, b)$ to node p, all lying in in a single neighborhood $N(a, b + r + 1)$. Furthermore, we have shown that any correct node $p(x, y)$ can hear all one-hop and two-hop messages, by round $(k+2)(\lfloor \frac{y-r-1}{r} \rfloor + 1)$. Node p can sample these messages over $k + 1$ rounds and, since the $O(\log^{(k)}(n) + k)$ Bad Santa strategy is used for selecting $S_{p,i}$, node p is guaranteed to receive a correct message. This completes the induction.

In terms of resource bounds, we can again consider the situations where p must deal with (either broadcasting or receiving) a message: (1) p receives messages in order to commit, (2) p broadcasts it has committed, and (3) p facilitates a two-hop message. We consider each case. To address (1), by the Bad Santa protocol, p listens to $O(\log^{(k)} n + k)$ messages in expectation. To address (2), note that p broadcasts that it has committed $k + 2 = O(k)$ times. To address (3), we note that p belongs to many B'_q sets for different nodes q; however, regardless of which B'_q set, p only ever has at most two sister nodes. Therefore, considering broadcast along the x and y coordinates, the number of sister nodes is $O(1)$; the number of

broadcasts due to two-hop paths messages is thus $O(k)$. The same arguments regarding fingerprints as given in the proof of Theorem 4 apply here which concludes the proof. This leads each node being awake over $O(\log^{(k)} n + k)$ time slots in expectation in Stage 2. Over both stages, each node sends $O(n \log^2 |m| + k|m|)$ bits, and listens to an expected $O(n \log^2 |m| + (\log^{(k)} n)|m|)$ bits. □

5.4 Unknown Start Time and Source(s)

In our previous protocols, *both the source of the message and the time the message was sent out needed to be pre-established.* Furthermore, our previous protocol allowed a savings on the fraction of required awake time only in Stage 2. The new protocol we present here assumes that every $2r+1$ by $2r+1$ square contains $t < \frac{n}{16+\epsilon}$ faults. Specifically, we require that no more than a $1/2 - \epsilon$ fraction of the nodes are faulty nodes in any $r/2$ by $r/2$ square. The benefits of this protocol are that it is 1) more energy efficient than using the protocol of [6]; 2) avoids the need to have the source and sending time pre-specified; and 3) reduces the awake time over the entire execution. Therefore, this algorithm is preferable when the circumstances of the fault model permit.

Let Q_i refer to the set of nodes in some $\frac{r}{2} \times \frac{r}{2}$ square in the grid. Our algorithm relies on correctly transmitting a message m from Q_{i-1} to Q_i where Q_{i-1} and Q_i are disjoint and neighboring squares i.e. the squares are neighbors abutting each other. Critical to our algorithm is an *assignment* of nodes in Q_{i-1} to nodes in Q_i. This assignment can be viewed as an undirected bipartite graph with the two disjoint sets of vertices being Q_{i-1} and Q_i and the assignment represented via edges. For $p \in Q_{i-1}$ and $q \in Q_i$, p listens to q and q listens to p *if and only if there is an edge between p and q in the bipartite graph.* This assignment is constructed such that all but a small fraction of correct nodes in Q_i receive a majority of correct messages from correct nodes in Q_{i-1} (and vice versa). This allows correct nodes in Q_i to majority filter on the messages they receive and decide upon the correct message. Thus, a message can be transmitted securely from $\frac{r}{2} \times \frac{r}{2}$ square to $\frac{r}{2} \times \frac{r}{2}$ square. For a message m and for any square Q_i to which m is sent by the above protocol, let $G(Q_i, m)$ be the set of correct nodes in Q_i that receive m after majority filtering over the accepted messages as described above. A result in [26] establishes the following theorem which we state without proof:

Theorem 9. *For any pair of squares Q_{i-1} and Q_i, there is non-zero probability of an assignment between nodes in Q_{i-1} and nodes in Q_i with the following properties:*

- *the degree of each node is at most a constant C which is independent of r,*
- *if $|G(Q_{i-1}, m)| \geq (1/2 + \epsilon/2)|Q_{i-1}|$, then $|G(Q_i, m)| \geq (1/2 + \epsilon/2)|Q_i|$.*

We will refer to an assignment with the two properties stated in Theorem 9 as a *robust assignment.* Although a method of assignment is not specified, Theorem 9 guarantees that a robust assignment must exist. We now consider the problem of find such an assignment.

Corollary 1. *A robust assignment between squares Q_{i-1} and Q_i can be found in time that is exponential in r^2.*

Proof. Consider any assignment between squares Q_{i-1} and Q_i as a bipartite graph G as described above. Both Q_{i-1} and Q_i have constant size $d = \frac{r^2}{4}$ so the number of possible bipartite graphs is at most $2^d \times 2^d = 2^{2d}$. Note that this is an upper bound on the number of different ways in which the faulty nodes can be placed in G. We know by Theorem 9 that there is non-zero probability that edges between Q_{i-1} and Q_i satisfy the property that a $(3/4 - \epsilon)$-fraction of the nodes in Q_i have a majority of correct neighbors in Q_{i-1}. Consider each of the at most 2^{2d} possible configurations of faulty nodes. For each such configuration, check whether all correct nodes in Q_i have at least a $(3/4 - \epsilon)$ fraction of correct neighbors in Q_{i-1}. By Theorem 5, such a configuration must exist and can be found by this exhaustive search which requires examining at most $2^{2d} = 4^{(r^2/4)}$ graphs. □

Therefore, for $r = \Theta(1)$, following the above procedure in Corollary 1 yields a robust assignment in constant time. Alternatively, a random regular graph will induce a desired assignment with probability at least $1 - 1/r^c$ for some constant $c > 0$. Recall that each node is assigned to C neighbors where C is

independent of r. Therefore, for a sufficiently large value $r = \Theta(1)$, nodes are listening to a small fraction of a square. Finally, we note that it is sufficient to find one robust assignment and use it for all pairs Q_{i-1} and Q_i.

5.4.1 Protocol Using Robust Assignment

We now describe and argue the correctness of a simple algorithm for reliable broadcast which we call ALG. ALG operates in stages of $\eta = r^2/4$ rounds. Over all rounds, each node that has committed to a message will broadcast at its scheduled slot. At every η^{th} round, a node enters into the listening state for one full round. That is, during this η^{th} round, all nodes are listening to all nodes in its $\frac{r}{2} \times \frac{r}{2}$ square *throughout the round*. At the end of this round, if a node p has received an identical message from a majority of nodes in its $2r + 1 \times 2r + 1$ square, p commits to this message.

For all other $\eta - 1$ rounds in a stage, a node sends and listens as dictated by a robust assignment and the broadcast schedule. That is, a node p listens to node q 1) if and only if p and q are assigned to each other under the robust assignment; and 2) when q is scheduled to broadcast. A robust assignment can be found as stated in Corollary 1 prior to deploying the radio network and this assignment can be preprogrammed into the nodes and used for all pairs of squares. Any node p may act as a source node. In this case, the source node will broadcast its message to its $r/2 \times r/2$ square in its time slot in an η^{th} round when all nodes are awake; the message should include a declaration that p is acting as a source. As in [8, 5, 6, 7], every node in the source's square commits to m and proceeds to broadcast m during their respective scheduled time slots. From this point, the message is propagated from $\frac{r}{2} \times \frac{r}{2}$ square to $\frac{r}{2} \times \frac{r}{2}$ square by sending and listening according to the robust assignment in a deterministic fashion: a square sends to the squares above and below and to the left and the right, in that order; Figure 7 illustrates this. Communication from one square to an adjacent square can be accomplished with a single round used per direction. Note that if η is not divisible by 4, then we simply interrupt on the special η^{th} round, and continue with the next direction afterwards. Therefore, a correct node will know this order and listen to its adjacent squares using the corresponding robust assignment in accordance with the broadcast schedule. As before, we may assume that the partitioning of the network into squares and the ordering and synchronization issues are dealt with through the nodes' internal programming; these details are outside the scope of this work. The exact propagation of a message depends on the robust assignment used and the behaviour of the faulty nodes; however, we can show correctness for the task of reliable broadcast.

By Theorem 9, at least a $(1/2 + \epsilon/2)$ fraction of correct nodes in every square will eventually receive identical messages from the majority of nodes to which it has been assigned. At this point, such a correct node can commit to a message and begin broadcasting, again according its robust assignment and scheduled time slots. Finally, we address the remaining fraction of correct nodes in a $r/2 \times r/2$ square that may not be able to commit to a message. Recall that at every η^{th} time slot, all nodes are listening for the entire round. Assuming that a $(1/2 + \epsilon/2)$-fraction of the correct nodes in the square have committed to the correct message, this allows the remaining fraction of correct nodes in a square to majority filter on incoming messages during this round and commit to the correct message.

In terms of costs, note that each node is always listening to at most C time slots in each of $\eta - 1$ rounds and listening to $r^2/4$ time slots in the η^{th} round; a total cost of $C(\eta - 1) + r^2/4$ over η rounds. Therefore, each node sends $O(|m|)$ bits per round and has an amortized cost of $O(C)$ time slots per round and an amortized cost of $O(C|m|)$ bits per round. Since C is a constant, this establishes our claims in Theorem 5.

5.5 Practical Considerations

We finish off this section by remarking on more practical considerations regarding our protocols. To start, we note that the grid model that we have adopted for applying our Bad Santa protocols is fairly flexible. Empty locations in the grid may correspond to failed nodes or simply the absence of a device altogether. The work in [6] generalizes results on reliable broadcast on the grid to arbitrary graphs where the problem is defined in terms of connectivity; our results easily generalize to such a setting and we refer the reader to [6] for more details. We also briefly mention that certain classes of random graphs may mapped to the grid model; the details depend on the type of random graph utilized. For example, if nodes are placed

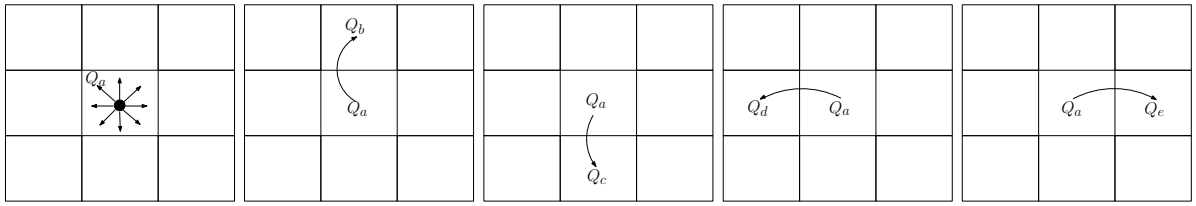

Figure 7: An illustration of the reliable broadcast protocol of Section 5.4. (a) Some node in Q_a acts as a source node to send a message to its square. Transmission from Q_a to (b) Q_b above, (c) Q_c below , (d) Q_d to the left and (e) Q_e to the right, using a robust assignment.

uniformly at random in the two-dimensional plane, we can partition the plane into a grid and then map nodes to their nearest intersection point to achieve the grid model. In general, so long as the number of faults in a neighborhood does not exceed $t < (r/2)(2r + 1)$, this mapping will work. We are simply sketching this idea for the interested reader; clearly, the details of how many nodes need to be dropped to guarantee at most t faults in any neighborhood (with sufficient probability) and how the broadcast radius should be defined are details that we leave to future work. We refer the interested reader to work in [10] which deals with issues of probabilistic failures in the grid model and a random network model. Next, we offer some discussion on the aspects of the storage and processing overhead incurred by our algorithms, as well as some exploration of the utility of our protocols in terms of bit complexity savings.

5.5.1 Storage and Processing Overhead

Recall that devices in the radio network are considered to be resource constrained. Here, we briefly discuss the costs associated with our algorithms in terms of processing and storage overhead, and we argue that these costs are reasonable. In particular, we argue that message storage and message processing costs are the primary costs. Note that these are costs that must be paid, to an even larger extent, in the original protocols of [5, 6]. Furthermore, we argue that these costs are negligible in comparison to the cost of sending/receiving; hence, our algorithms do indeed achieve a power savings.

In terms of storing data, consider our protocols where the send time and location of the source is known. Each node must store information on the current time slot, the slot when it can broadcast, its location relative to the source, its set of neighbors in the broadcast region, and information on the type of Bad Santa protocol being used (i.e. number of streams, the current stream, which time slots it is listening to); all of these can be stored with a small amount of overhead. The protocol for the case where we have Byzantine faults also requires storage of fingerprints, which are small compared to an actual message, and a hash function. The use of hash functions for such resource constrained environments has been tested in [27] and in [28] (on the MICA series); it appears the storage costs are no obstacle. Therefore, the main storage overhead in our algorithms appears to result from messages. The length of these messages is likely application dependent and memory sizes can differ with the device in question. In [29], the MICA2 devices are stated to have 4KB of memory. However, we note that current memory sizes on these radio network devices can be sizable. For instance, in [30], the authors report that a flash-memory of 32KB and the ability to add an additional storage capacity (up to 1GB) for the devices studied. Therefore, memory size can be chosen for the application and corresponding message sizes in question; but regardless, we do not anticipate that our algorithms incur an unreasonable amount of storage overhead over what is needed for storing messages.

The main processing cost of our algorithms differs per case. For the Byzantine fault-tolerant algorithm of Section 5.2, the main cost is likely due to the use of the hash function. Recall that this operation must be done fairly frequently in order for a node to commit to the correct message. While we do use a hash function, we note that we don't use public key cryptography in any of our algorithms which has generally been considered to be expensive for power constrained nodes due to the need for sending, receiving, and storing public keys and executing encrypt/decrypt operations [31]. More sophisticated techniques

are now available which require less energy; however, the costs are still quite high. For instance, in [31], measurements by the authors using the MICA2DOT unit demonstrate a cost of 2302.70 mWs (microwatt seconds) and 53.70 mWs for 2048-bit RSA signature generation and verification, respectively. Elliptic Curve Crytography (ECC) is a popular alternative to RSA since it has smaller key sizes. For 224-bit ECC, the same authors measure costs with the MICA2DOT unit at 61.54 mWs and 121.98 mWs for signature generation and verification, respectively. Both RSA-2048 and ECC-224 are recommended by RSA Security as the new standard in order to protect data past the year 2010 [32]. These costs should be compared to the cost of broadcast in on the Lucent IEEE 802.11 2Mbps WaveLAN PC Card which is measured at 266 mWs. Therefore, it is not clear that an algorithm could claim to save significant power by employing full cryptographic schemes.

On the other hand, hash functions for power constrained environments have been considered in the literature and it appears the processing costs are reasonable [27]. In particular, the SHA-1 hash function can be applied with very little power consumption; again with the MICA2DOT unit, the cost is measured to be 5.9 μWs/byte is measured in [32]. Therefore, hasing a 1Kb message would incur 5.9 mWs; notably this is far less than the cost of sending or receiving.

For the algorithm of Section 5.4, the most significant processing costs would appear to arise from the need to majority filter on all incoming messages; however, such a comparison operation is certainly feasible in radio network devices. Finally, for the fail-stop case, the algorithm of Section 5.1 does not need to apply a hash function to messages and we do not anticipate significant processing costs here. In some cases, additional processing overhead will come from comparing hashes and accessing a random number generator; however, we anticipate that these additional processing overheads will be small in comparison to the cost of storing and processing messages.

5.5.2 Saving on Bit Complexity

Recall that our Byzantine fault-tolerant reliable broadcast protocol of Section 5.2 achieves asymptotically lower bit complexity through the use of hashing. However, there is the question of when such savings would be seen in practice. Packet sizes are discrete, and in many cases, the hash of a message may require the same number of packets as sending the message itself. If messages are small, then the bit complexity savings achieved by our protocol will be consequently smaller. However, we note that if messages are sizable then there is a benefit to the hashing technique.

In the face of large amounts of data collection and querying, data aggregation techniques have been proposed to reduce the overall communication costs since processing is generally less costly than sending data (see [33, 34] for more on this). Despite these techniques, there are applications for wireless networks that require transmission of large amounts of data even after processing. For instance, surveillance applications that require sending significant amounts of data have been proposed involving image and video data [35] such as in traffic monitoring [36] and transmitting biometric data in security scenarios [37] where image data must be sent over wireless networks. Therefore, there are indeed applications where large messages might be transmitted and we anticipate more such situations will arise in the future. Under such scenarios, we would expect our algorithms to save substantially on bit complexity.

When considering large messages, there is also the issue of slot size to consider. Modifying time division multiple access (TDMA) has been considered (see [38]) and it is possible that similar proposals could be used to allow large messages to be sent within a single time slot without underutilizing bandwidth. Alternatively, time slots could be reset by the dealer in order to accomodate large future transmissions; the details of this would likely be application specific and we leave this as a topic for future work.

6 Future Work and Conclusion

We have designed new algorithms for achieving significant energy savings in radio networks. To achieve these ends, we have defined and analyzed a novel data streaming problem which we call the Bad Santa problem. We have shown how our results on this problem can be applied to the problem of reliable broadcast in a grid radio network. Our algorithms for reliable broadcast on a grid consume significantly less power than any other algorithms for this problem of which we are aware.

Several open problems remain including: Can we close the gap between the upper and lower-bound for the multi-round Bad Santa problem? Can we achieve more energy efficiency for the optimal number of Byzantine faults? Can we tolerate more faults for the fail-stop model and still be energy efficient? Can we tolerate more faults in the unknown source and message time scenario? Can we generalize our techniques to radio networks that are not laid out on a two dimensional grid (perhaps classes of random graphs)? Are there other applications for the Bad Santa problem both in and outside the domain of radio networks?

Acknowledgements: We gratefully thank Kui Wu, Chiara Petrioli and James Horey for their helpful comments on issues of power consumption in radio networks. We are also indebted to the anonymous reviewers, particularly Reviewer 2, for their helpful comments.

References

[1] Qin Wang, Mark Hempstead, and Woodward Yang. A Realistic Power Consumption Model for Wireless Sensor Network Devices. In 3^{rd} *Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2006.

[2] Laura Marie Feeney and Martin Nilsson. Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment. In *INFOCOM*, 2001.

[3] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David E. Culler, and Kristofer S. J. Pister. System Architecture Directions for Networked Sensors. In 9^{th} *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 93–104, 2000.

[4] Lan Wang and Yang Xiao. A Survey of Energy-Efficient Scheduling Mechanisms in Sensor Networks. *Mobile Networks and Applications*, 11:723–740, 2006.

[5] Vartika Bhandari and Nitin H. Vaidya. On Reliable Broadcast in a Radio Network. In 24^{th} *Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 138–147, 2005.

[6] Vartika Bhandari and Nitin H. Vaidya. On Reliable Broadcast in a Radio Network: A Simplified Characterization. Technical report, CSL, UIUC, May 2005.

[7] Chiu-Yuen Koo, Vartika Bhandhari, Jonathan Katz, and Nitin Vaidya. Reliable Broadcast in Radio Networks: The Bounded Collision Case. In 25^{th} *Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 258 – 264, 2006.

[8] Chiu-Yuen Koo. Broadcast in Radio Networks Tolerating Byzantine Adversarial Behavior. In 23^{rd} *Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 275–282, 2004.

[9] Douglas R. Stinson. *Cryptography: Theory and Practice,* 3^{rd} *Ed.* Chapman & Hall, 2006.

[10] Vartika Bhandari and Nitin H. Vaidya. Reliable Broadcast in Wireless Networks with Probabilistic Failures. In *INFOCOM*, pages 715–723, 2007.

[11] Seth Gilbert, Rachid Guerraoui, and Calvin C. Newport. Of Malicious Motes and Suspicious Sensors: On the Efficiency of Malicious Interference in Wireless Networks. In *International Conference On Principles Of Distributed Systems (OPODIS)*, pages 215–229, 2006.

[12] Gregory Chockler, Murat Demirbas, Seth Gilbert, Calvin Newport, and Tina Nolte. Consensus and Collision Detectors in Wireless Ad Hoc Networks. In 24^{th} *Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 197 – 206, 2005.

[13] Evangelos Kranakis, Danny Krizanc, and Andrzej Pelc. Fault-Tolerant Broadcasting in Radio Networks. *Journal of Algorithms*, 39(1):47–67, 2001.

[14] Andrzej Pelc and David Peleg. Broadcasting with Locally Bounded Byzantine Faults. *Information Processing Letters*, 93(3):109–115, 2005.

[15] Leszek Gąsieniec, Erez Kantor, Dariusz R. Kowalski, David Peleg, and Chang Su. Time Efficient k-shot Broadcasting in Known Topology Radio Networks. *Distributed Computing*, 21(2):117–127, 2008.

[16] Monika Henzinger, Prabhaker Raghavan, and Sridar Rajagopalan. Computing on Data Streams. Technical Report SRC-TN-1998-011, Digital Systems Research Center, 1998.

[17] S. Muthukrishnan. *Data Streams: Algorithms and Applications*. Now Publishers Inc, 2005.

[18] Ian Munro and Mike Paterson. Selection and Sorting with Limited Storage. *Theoretical Computer Science*, pages 315–323, 1980.

[19] Sudipto Guha and Andrew McGregor. Approximate Quantiles and the Order of the Stream. In *ACM Symposium on Principles of Database Systems*, pages 273–279, 2006.

[20] Sudipto Guha and Andrew McGregor. Lower Bounds for Quantile Estimation in Random-Order and Multi-Pass Streaming. In 34^{th} *International Colloquium on Automata, Languages and Programming*, 2007.

[21] Noga Alon, Yossi Matias, and Mario Szegedy. The Space Complexity of Approximating the Frequency Moments. In 28^{th} *Annual ACM Symposium on Theory of Computing (STOC)*, pages 20–29, 1996.

[22] Eric Demaine, Alejandro López-Ortiz, and Ian Munro. Frequency Estimation of Internet Packet Streams with Limited Space. In 10^{th} *Annual European Symposium on Algorithms (ESA)*, pages 348–360, 2002.

[23] Graham Cormode, Flip Korn, S. Muthukrishnan, and Divesh Srivastava. Space- and Time-Efficient Deterministic Algorithms for Biased Quantiles Over Data Streams. In 25^{th} *ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 263 – 272, 2006.

[24] Valerie King, Cynthia Phillips, Jared Saia, and Maxwell Young. Sleeping on the Job: Energy-Efficient and Robust Broadcast for Radio Networks. In 27^{th} *ACM symposium on Principles of Distributed Computing (PODC)*, pages 243–252, 2008.

[25] Andrew Yao. Probabilistic computations: Toward a Unified Measure of Complexity. In 18^{th} *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 222–227, 1977.

[26] Jared Saia and Maxwell Young. Reducing Communication Costs in Robust Peer-to-Peer Networks. *Information Processing Letters*, 106(4):152–158, 2008.

[27] Kaan Yüksel, Jens-Peter Kaps, and Berk Sunar. Universal Hash Functions for Emerging Ultra-Low-Power Networks. In *Communications Networks and Distributed Systems Modeling and Simulation Conference (CNDS)*, page n. pag., 2004.

[28] HangRok Lee, YongJe Choi, and HoWon Kim. Implementation of TinyHash Based on Hash Algorithm for Sensor Network. In *World Academy of Science, Engineering and Technology (WASET)*, pages 135–139, 2005.

[29] Nathan Cooprider, Will Archer, Eric Eide, David Gay, and John Regehr. Efficient Memory Safety for TinyOS. In *Sensys'07: ACM International Conference on Embedded Networked Sensor Systems*, pages 205–218, 2007.

[30] Demetrios Zeinalipour-Yazti, Som Chandra Neema, Vana Kalogeraki, Dimitrios Gunopulos, and Walid Najjar. Data Acquisition in Sensor Networks with Large Memories. In 21^{st} *International Conference on Data Engineering Workshops*, pages 1188–1188, 2005.

[31] Krzysztof Piotrowski, Peter Langendoerfer, and Steffen Peter. How Public Key Cryptography Influences Wireless Sensor Node Lifetime. In *Fourth ACM Workshop on Security of Ad Hoc and Sensor Networks*, pages 169–176, 2006.

[32] Arvinderpal S. Wander, Nils Gura, Hans Eberle, Vipul Gupta, and Sheueling Chang Shantz. Energy Analysis of Public-Key Cryptography for Wireless Sensor Networks. In 3^{rd} *International Conference on Pervasive Computing and Communications*, pages 324–328, 2005.

[33] Johannes Gehrke and Samuel Madden. Query Processing in Sensor Networks. *IEEE Pervasive Computing*, 3(1):46–55, 2004.

[34] Mohamed Watfa, William Daher, and Hisham Al Azar. A Sensor Network Data Aggregation Technique. *International Journal of Computer Theory and Engineering*, 1(1):19–26, 2009.

[35] Edmund Y. Lam, King-Shan Lui, and Vincent W. L. Tam. Image and Video Processing in Wireless Sensor Networks. *Multidimensional Systems and Signal Processing*, 20(2):99–100, 2009.

[36] Jiang Yu Zheng and Shivank Sinha. Line Cameras for Monitoring and Surveillance Sensor Networks. In 15^{th} *International Conference on Multimedia*, pages 433–442, 2007.

[37] Rajani Muraleedharan, Lisa Ann Osadciw, and Yanjun Yan. Resource optimization in Distributed Biometric Recognition Using Wireless Sensor Networks. *Multidimensional Systems and Signal Processing*, 20(2):165–182, 2009.

[38] Ted Herman, , and Sébastien Tixeuil. A Distributed TDMA Slot Assignment Algorithm for Wireless Sensor Networks. *Algorithmic Aspects of Wireless Sensor Networks*, 3121:45–58, 2004.